VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Návrh a implementace prohlížeče reálných a historických trendů v prostředí .NET pro účely průmyslové automatizace

Design and Implementation of real and Historical Trend Viewer in .NET Environment for Industrial Automation Area

2012 Vít Otevřel

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Vít Otevřel**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Návrh a implementace prohlížeče reálných a historických trendů v prostředí .NET pro účely průmyslové automatizace
Design and Implementation of real and Historical Trend Viewer in .NET Environment for Industrial Automation Area

Zásady pro vypracování:

Důraz je kladen především na rychlost, jednoduchost a praktičnost ovládání. Předpokládá se přítomnost standardních funkcí prohlížečů (zoom, kurzory, konfigurace os, atd.), pokročilých funkcí (aplikace jednoduchých filtrů, statistických a matematických funkcí na zobrazený signál) ale také schopnost integrace s dalšími moduly rozšiřujícími funkcionalitu prohlížeče (např. modul identifikace soustav).

body zadání:
1. Zpracování teoretických možností poskytovaných platformou .NET pro účely vizualizace trendů v průmyslové automatizaci.
2. Návrh prohlížeče reálných a historických trendů v prostředí .NET pro účely průmyslové automatizace.
3. Implementace navrženého řešení pro vybraný modelový případ nasazení v řídicích systémech.
4. Diskuze dosažených výsledku. Stanovení základních limitů realizované aplikace.
5. Zhodnocení vlastní práce

Seznam doporučené odborné literatury:

1. KAČMÁŘ, D. *Programujeme .NET aplikace ve Visual Studiu .NET.* Vyd. 1. Brno:Computer Press, 2001. 344 s. ISBN 80-7226-569-5.
2. HERNANDEZ, M. J. - VIECAS, J. L. *Knihovna programátora: myslíme v jazyku SQL:tvorba dotazů.* 1. vyd. Praha: Grada Publishing, 2004. 378 s. ISBN 80-247-0899-X .

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Jiří Koziorek, Ph.D.**

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

„Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

V Ostravě dne 4. května 2012 podpis 

Abstrakt

Tématem této diplomové práce je navrhnout a implementovat prohlížeč reálných a historických trendů v prostředí .NET. Cílem je vytvořit prohlížeč se standardními funkcemi trendových prohlížečů (zoom, kurzory, konfigurace os, atd.), který by měl ovšem zvládnout i pokročilejší funkce (aplikace jednoduchých filtrů, statistických a matematických funkcí na zobrazený signál). Další vlastností prohlížeče by měla být schopnost integrace s dalšími rozšiřujícími moduly, pro rozšíření funkcionality prohlížeče (např. modul identifikace soustav, moduly pro práci s dalšími zdroji dat).

Abstract

This master work deal is design and implementation of real and historical trend viewer in .NET environment. The aim of this work is create a viewer with standard functions (zoom, cursor, axis configuration, etc.), of course this viewer should be manage with advanced functions (simple filters, statistic and mathematic function on display signal). Other feature of viewer should be ability with other extended modules for extend a viewer functionality (for example, module for system identification, modules for operations with other data sources).

Klíčová slova

.NET framework, historický prohlížeč, reálný prohlížeč, trendy.

Key words

.NET framework, historical viewer, real viewer, trends.

Seznam použitých symbolů a zkratek

.NET	Softwarová technologie, platforma Microsoftu
API	Application Programming Interface
ASP	Active Server Pages
BCL	Base Class Library
CLR	Common Language Runtime
CPU	Central Processing Unit
CSV	Soubor určený pro výměnu tabulkových dat (Comma-separated values)
DBF	dBase databáze
DLL	Dynamic Link Library
FFT	Fast Fourier Transformation
GPL	GNU General Public License
GUI	Graphical User Interface
HMI	Human Machine Interface
ID	Identifikátor
IDE	Integrated Development Environment
MDI	Multiple Document Interface
MIT	Massachusetts Institute of Technology License
ODBC	Open Database Connectivity
OOP	Objektově orientované programování
PC	Počítač (Personal Computer)
PDF	Přenosný formát dokumentů (Portable Document Format)
PLC	Programmable Logic Controller
SDI	Single Document Interface
SQL	Structured Query Language
TDI	Tabbed Document Interface
TXT	Textový soubor
VB	Visual Basic
XLS	Přípona souborů vytvořených v aplikaci Microsoft Excel
XML	Extensible Markup Language

Poděkování

Zde bych rád poděkoval všem, kteří mi pomohli při tvorbě práce a rodině za podporu ve studiu.

Obsah

1 Úvod

Na trhu trendových prohlížečů existuje celá řada programů s různými funkcemi. Tyto programy mohou být rozděleny do dvou základních skupin – prohlížeče historických a reálných trendů, oba typy aplikací umožňují zobrazit data z datových zdrojů do grafické podoby – grafu.

Cílem diplomové práce je vytvoření funkčního trendového prohlížeče. Toto zadání bylo zadáno firmou *Ingeteam*, která bude aplikaci využívat. Hlavním důvodem vývoje vlastního trendového prohlížeče ve spolupráci s firmou *Ingeteam* jsou ceny profesionálních trendových prohlížečů, které bývají dost drahé, a oproštění tak od licenčních podmínek.

Téma trendového prohlížeče je velmi rozsáhlé, proto byl jeho vývoj rozdělen na tři části – mezi tři studenty, tzn. byly vypsány 3 diplomové práce. Jedna z prací se zabývá vývojem knihoven pro přístup k jednotlivým datovým zdrojům (CSV, PLC, …). Další práce je věnována vývoji knihovny či zásuvného modulu, který by měl identifikovat soustavu z vybraných křivek zobrazených v grafu prohlížeče. Tato diplomová práce je zaměřena na návrh a implementaci prohlížeče historických a reálných trendů v prostředí .NET. Úkolem je navrhnout GUI celého prohlížeče a implementovat standardní funkce jako například zobrazování jednotlivých dat – signálů z datových zdrojů do grafické podoby – grafu, kurzory, zoom, nastavení prohlížeče a jednotlivých os grafu. Aplikace je rozdělena na tři části, z tohoto důvodu je potřeba se domluvit s ostatními kolegy na tom, jak předávat informace mezi GUI a knihovnami – vytvořit komunikační rozhraní, kterému je věnována podkapitola v rámci třetí kapitoly. Práce je rozdělena do několika kapitol.

Druhá kapitola obsahuje zpracování teoretických možností poskytovaných platformou .NET pro účely vizualizace trendů v průmyslové automatizaci. Hlavní obsahem kapitoly je podrobné seznámení se s požadavky, které jsou kladeny na reálné a historické trendové prohlížeče. Dalším bodem kapitoly je analýza několika prohlížečů, které jsou již na trhu. Úkolem této analýzy bylo seznámení se s řešením trendových prohlížečů a vytvoření nadhledu o práci a funkčnosti těchto prohlížečů.

Obsahem třetí kapitoly je návrh prohlížeče historických a reálných trendů v prostředí .NET pro účely průmyslové automatizace. V této kapitole je blíže popsána platforma .NET, programovací jazyk C# a výběr jednotlivých komponent pro prohlížeč. Důležitým bodem této kapitoly je návrh vlastního prohlížeče, který je inspirován analýzou z druhé kapitoly.

Čtvrtá kapitola se zabývá implementací navrženého řešení pro vybraný modelový případ nasazení v řídicích systémech. V dalších podkapitolách jsou probrány využití a implementace jednotlivých .NET komponent použitých při realizaci trendového prohlížeče. Tato kapitola obsahuje i popis některých použitých algoritmů.

Poslední kapitolou práce je závěr, ve kterém je zhodnocena celá práce. Shrnutí co se povedlo realizovat a jak vypadá vlastní výsledný produkt – prohlížeč reálných a historických trendů.

2 Analýza trendových prohlížečů

Co se vlastně myslí tím, když se mluví o trendech nebo trendových prohlížečích? Trendy jsou vlastně závislosti určitých, námi zvolených veličin na čase, které jsou zobrazeny v grafické podobě. Obecně trendovým prohlížečem je program, pomocí kterého lze zobrazovat aktuální nebo uložená data v grafu.

2.1 Rozdělení trendových prohlížečů

Hlavním úkolem trendových prohlížečů je zobrazení dat v grafické podobě – grafu. Trendové prohlížeče, ale nemusí zobrazovat pouze data v grafu, ale mohou umožnit i jejich náhled formou tabulky.

Rozdělení trendových prohlížečů:

- Historické trendy.
- Reálné trendy.

Programy, které jsou zařazeny do kategorie historických trendů jsou aplikace zobrazující starší data, již uložená v datových zdrojích (např. data uložená v souborech různých formátů nebo databázích), které byly během provozu zaznamenány. Do druhé kategorie, reálných trendů, spadají programy, které dokáží zobrazovat data on-line – tzn. aktuální data. Aplikace zobrazující reálné trendy obvykle komunikují s určitým typem hardwaru (např. PLC), ze kterého aplikace vyčítají data a následně je zobrazují do grafu periodicky (např. každou sekundu).

Mezi prohlížeče historických trendů patří programy jako *TrendAnalyzer* [8] nebo *IbaAnalyzer* [10]. Oba zmíněné programy zvládají stejnou práci – zobrazují data do grafu a navíc umožňují provádět jednoduché analýzy se zobrazenými daty.

Jako zástupce reálných trendových prohlížečů je zmíněna komponenta poměrně rozsáhlého vizualizačního prostředí *Siemens WinCC* [11], tato komponenta se nazývá *Trend Control*. Komponenta umožňuje kromě aktuálních dat načtených z PLC zobrazovat i historické data.

2.2 Požadavky na trendový prohlížeč

Protože byl vyvíjen tento prohlížeč pro firmu *Ingeteam*, na začátku vývoje bylo sepsáno několik základních požadavků, které by měl trendový prohlížeč splňovat.

Požadavky zadávající firmy *Ingeteam*:

- Programovací jazyk C#.
- Graf.
- Kurzory.

- Správce os.
- Správce signálů.
- Správce zdrojů dat.
- Postprocesing – možnost další práce s daty (součet signálu, jednoduché matematické operace).
- Uložení konfigurace do XML [6] souboru.

Programovací jazyk C#

Jedním z prvních požadavků byla volba programovací platformy a programovacího jazyku. Prohlížeč byl vyvíjen pro platformu .NET a byl použit programovací jazyk C#. Tento jazyk byl zvolen z toho důvodu, aby softwaroví vývojáři *Ingeteamu*, kteří programují v jazyce C#, mohli v budoucnu doprogramovat např. další knihovnu či modul pro ještě neimplementovaný datový zdroj.

Graf

Pro trendové prohlížeče je zásadním, tedy nejdůležitějším prvkem grafické zobrazení dat – graf. Graf umožňuje graficky zobrazit závislosti několika veličin na čase nebo vzájemné závislosti určitých veličin.

Kurzory

Uživatel, který pracuje s trendovým prohlížečem někdy potřebuje zjistit hodnoty na všech křivkách v určitém čase, který je vynesen na ose X. Právě pro tuto činnost jsou vhodné kurzory, někdy též nazývané jako ukazatele či z angličtiny markery. Kurzor je vlastně čára protínající všechny křivky zobrazené v grafu. V tomto časovém okamžiku, kdy kurzor protíná křivky, by zde měla být i možnost zobrazit hodnoty z křivek.

Správce os

Pro trendový prohlížeč by bylo vhodné, aby bylo umožněno uživateli konfigurovat osy grafu. Tzn. uživatel by měl mít možnost upravovat některé vlastnosti os jako jsou popisky, rozsah os nebo také krok či interval pomocné mřížky. Požadavkem firmy *Ingeteam* bylo použití dvou os X a několika os Y. Počet os Y byl omezen na 8, 4 osy na pravé a 4 osy na levé straně grafu. Omezení počtu os Y je zde namístě, protože při větším počtu os se rapidně zmenšuje plocha grafu pro zobrazení křivek, což vede k nepřehlednosti zobrazovaných dat.

Správce signálů

Signálem je zde myšleno pole dat v datovém zdroji, přiřazené určité křivce. Pokud bude otevírán datový zdroj, ve kterém je uloženo více signálů, pak je důležité zobrazit všechny signály obsažené v datovém zdroji do přehledné tabulky. Tato tabulka by měla umožňovat jednoduchou konfiguraci signálu pro zobrazení v grafu, tím je myšlena změna parametrů signálu jako je barva, šířka nebo symbol bodu křivky grafu.

Správce zdrojů dat

Zdrojem dat je myšleno v jakém formátu jsou data uložena. Uživatel by měl mít možnost si vybrat z jakého zdroje budou data načítány.

Zdroji dat jsou myšleny některé z těchto známých formátů:

- Off-line:
    - CSV soubor.
    - SQL server.
    - ODBC.
    - DBF.
    - XLS.
    - MDF.
    - Binární soubor.
- On-line:
    - PLC Siemens S7.
    - Beckhoff Embedded PC.

Konfigurace uložená do XML [6]

Důležitým požadavkem je také ukládání konfigurace prohlížeče do souboru. Pro ukládání konfigurace byl zvolen soubor XML [6]. Uložení konfigurace je důležité, protože ukončíme-li práci s prohlížečem, uloží se rozložení prvků okna prohlížeče právě do tohoto konfiguračního XML [6] souboru. Dalším využitím bylo uložení konfigurace tabulky signálů do XML [6] souboru.

Postprocesing

Prohlížeč by měl zvládnout jednoduché matematické operace s vykreslenými křivkami. Jedná se hlavně o tyto operace: sčítání a odčítání křivek, zjištění průměru, minima a maxima.

2.3 Analýza trendových prohlížečů na trhu

Jelikož je na trhu s trendovými prohlížeči několik programů, bylo potřeba některé prohlížeče vyzkoušet a inspirovat se, jak některé funkce fungují a jak jsou vyřešeny.

Programy pracující s off-line daty:

- *TrendAnalyzer* [8].
- *Orion-M Data Viewer* [9].
- *ibaAnalyzer* [10].

Komponenta vizualizačního balíku *Siemens WinCC* [11], pracující s daty z PLC:

- Komponenta *Trend Control*.

Byly odzkoušeny pouze programy pracující s off-line daty, *TrendAnalyzer* [8], *Orion-M Data Viewer* [9] a *ibaAnalyzer* [10]. Tyto programy byly vyzkoušeny jako demo verze, tudíž byly některé jejich funkce omezené.

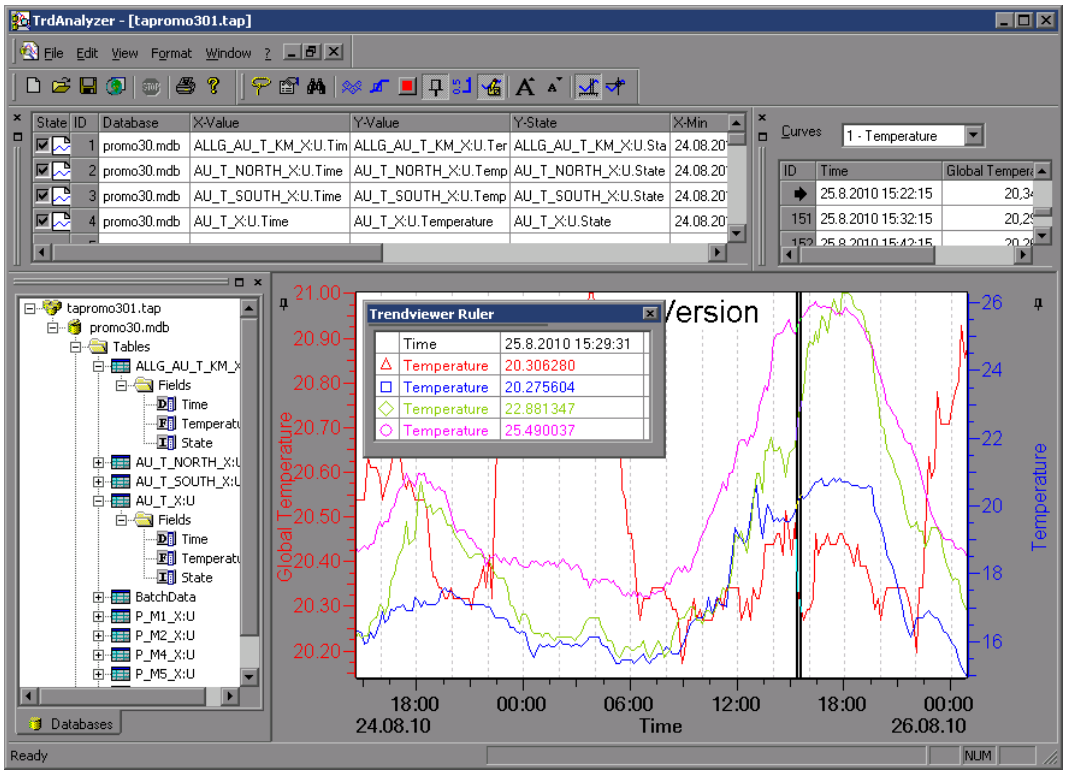
Zbylý program, respektive komponentu *Trend Control*, nebylo možno odzkoušet, ale naskytla se možnost vidět funkci této komponenty ve firmě *Ingeteam*.

2.3.1 TrendAnalyzer

Jedná se o program německé firmy *ICS GmbH*. *TrendAnalyzer 3.0* [8] je softwarový nástroj pro rychlé a jednoduché zobrazení, analýzu a dokumentaci naměřených hodnot. Program dokáže pracovat s několika formáty dat, jako jsou databáze (Access, dBase), CSV soubory, textové soubory nebo tabulky programu *Excel* a několika dalšími. [8]

Firma vyvíjející tento program využívá pro zobrazování křivek v grafu vlastní komponentu nazvanou *TrendViewer*. Tato komponenta je vyvíjena pod platformou .NET. [8]

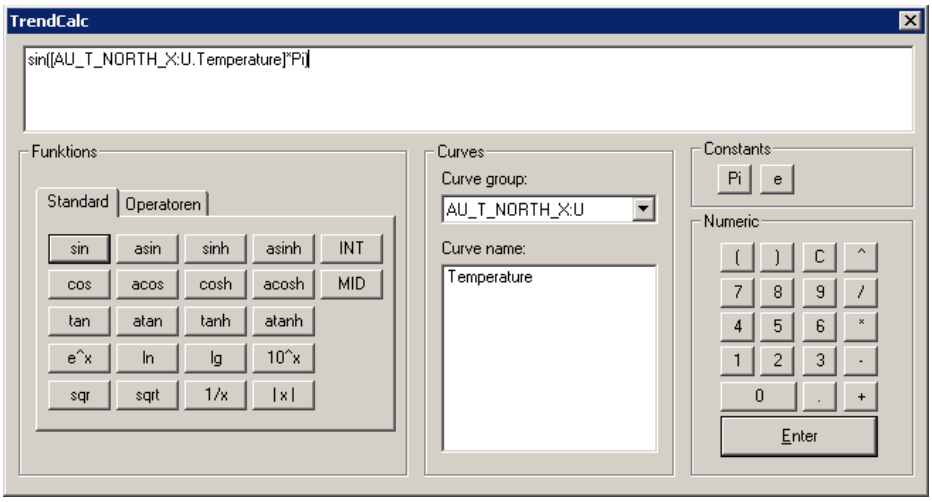
Program je placený a je možno jej zakoupit s 5 různými licencemi. Bez licence je program možno pouze otestovat jako demoverzi s přiloženými databázemi. Tudíž byla některá funkcionalita programu nedostupná, např. možnost použití vlastních dat. [8]



Obr. 1: TrendAnalyzer [8], zobrazení zkušebních dat

Jak je vidět z *Obr. 1*, grafické prvky jsou konfigurovatelné, a je možné je přesouvat v rámci okna aplikace. Po prvním spuštění programu jsou okna logicky uspořádané. [8]

Vlevo vedle grafu, v ukotveném plovoucím okně „Databases“ je zobrazen obsah datového zdroje. Jsou zde zobrazeny všechny signály obsažené v datovém zdroji. Okno „Databases“ slouží jako správce datových zdrojů. Tyto signály lze tahem myší přesouvat do seznamu křivek, což je v horní části aplikace, okno nazvané „Configuration“ (viz *Obr. 1*). Křivky, které jsou přidány do seznamu mohou být konfigurovány – mohou být měněny počáteční a koncové hodnoty (kde má křivka v grafu začínat a končit), dále je možno přiřadit křivce osu X a osu Y a v neposlední řadě určit zda má být křivka zobrazena v grafu. Aplikace nabízí výběr ze dvou os X a až 32 os Y. Tabulka křivek, je ale omezena na 31 řádků, což znamená, že může být zobrazeno najednou až 31 křivek. Pomocí pravého tlačítka myši mohou být z tabulky křivek jednotlivé křivky odebírány. Další funkcí dostupnou přes nabídku pravého tlačítka je tzv. „Trend Calculator“, zobrazený na *Obr. 2*, toto okno je zobrazeno jako modální dialog. [8]



Obr. 2: Trend Calculator - matematické operace s křivkami

Kalkulačka umožňuje provádět pouze matematické operace s jednou z křivek zobrazenou v grafu, tzn. nelze provádět operaci se dvěma křivkami, např. součet dvou křivek. Jak je vidět na *Obr. 2*, v kalkulačce je implementováno mnoho matematických operací, které mohou být s křivkou prováděny. Tlačítkem „Enter“ je potvrzována matematická operace a do seznamu křivek a grafu je přidána virtuální křivka s výsledkem matematické operace. [8]

Zobrazování a přibližování křivek bylo rychlé, ovšem vzhledem k tomu, že testovací data měly pouze kolem 250 vzorků na křivku, nebyla rychlost otestována objektivně. Použitá grafická komponenta umožňuje jak přibližování všech křivek, tak přibližování jednotlivých os zvlášť. Pro přibližování a posouvání jednotlivých os bylo použito klávesových zkratek a rolování kolečkem myši. Zajímavou funkcí komponenty je možnost přenášení os Y z jedné strany na druhou, čímž se zvyšuje konfigurovatelnost. [8]

Přes nabídku „Edit > Properties“ je dostupné nastavení v podobě modálního dialogového okna. Nastavení je rozčleněno do několika záložek, které seskupují jednotlivé části nastavení – grafu, křivek a os. [8]

Nastavení grafu obsahuje klasické položky jako jsou barvy grafu, nastavení použitého písma – velikost, barva, styl, nastavení pomocné mřížky – krok, velikost, šířka a barva. [8]

U nastavení křivek jsou zobrazeny pouze ty křivky, které jsou aktuálně zobrazeny v grafu. Konfigurace je přehledně zobrazena v tabulce, samozřejmostí je možnost nastavovat barvu, tloušťku a styl křivky. Z dalších nastavení je možnost zobrazení obálky křivky. Poslední možností je zde definování dvou limit křivky. Tyto limity mohou být pouze zobrazeny a definována jejich hodnota na ose Y. [8]

Jednotlivé osy, které jsou aktuálně využívány (X i Y) lze různě nastavovat. Mezi nastavení os patří barva osy, typ (lineární, časová), viditelnost osy v grafu, krok a šířka linky. Konfigurovatelnost os Y je rozšířena o volbu logaritmického zobrazení a nastavení rozsahu – minima a maxima. [8]

Program umožňuje pracovat pouze s jedním vloženým kurzorem – horizontálním nebo vertikálním. Hodnoty získané kurzorem z křivek jsou přehledně zobrazeny v plovoucím okně, které lze ukotvit kdekoliv v okně programu. Pomocí vertikálních kurzorů jsou zobrazovány hodnoty z křivek, zatímco horizontální kurzory zobrazují hodnotu z jednotlivých os Y. [8]

Je zde možnost prohlížet data křivek – na *Obr. 1* to je ukotvené plovoucí okno nazvané „Curves“. V tabulce mohou být zobrazeny pouze data těch křivek, které jsou zobrazeny v grafu. Data lze přepisovat a změny se ihned projeví v grafu. [8]

Zhodnocení

Vzhledem k tomu, že je program potřeba koupit, je dobře konfigurovatelný, v demo verzi nebylo možno vyzkoušet úplně všechny možnosti programu jako například import CSV souborů s vlastními daty (umožňuje i import ODBC). Jelikož je demo verze dodávána s několika testovacími databázemi, bylo testování prováděno pouze na těchto testovacích datech. Bylo by zajímavé otestovat rychlost vykreslování s více vzorky na křivku než bylo v testovacích souborech. V plné verzi je také možnost zobrazování fázových diagramů.

Výhody

- Množství podporovaných formátů (Excel, Access a dBase databáze, CSV, TXT).
- Podrobné nastavení křivek a os (možnost výběru až z 32 os Y a 2 os X).
- Možnost zobrazit a konfigurovat současně až 31 křivek.
- Matematické operace s křivkami – kalkulačka.
- Umožňuje vyhledávat naměřené hodnoty.

Nevýhody

- Aktuálně zobrazený pouze jeden kurzor – buď vertikální nebo horizontální.
- Nemožnost provádět matematické operace se dvěma křivkama najednou.

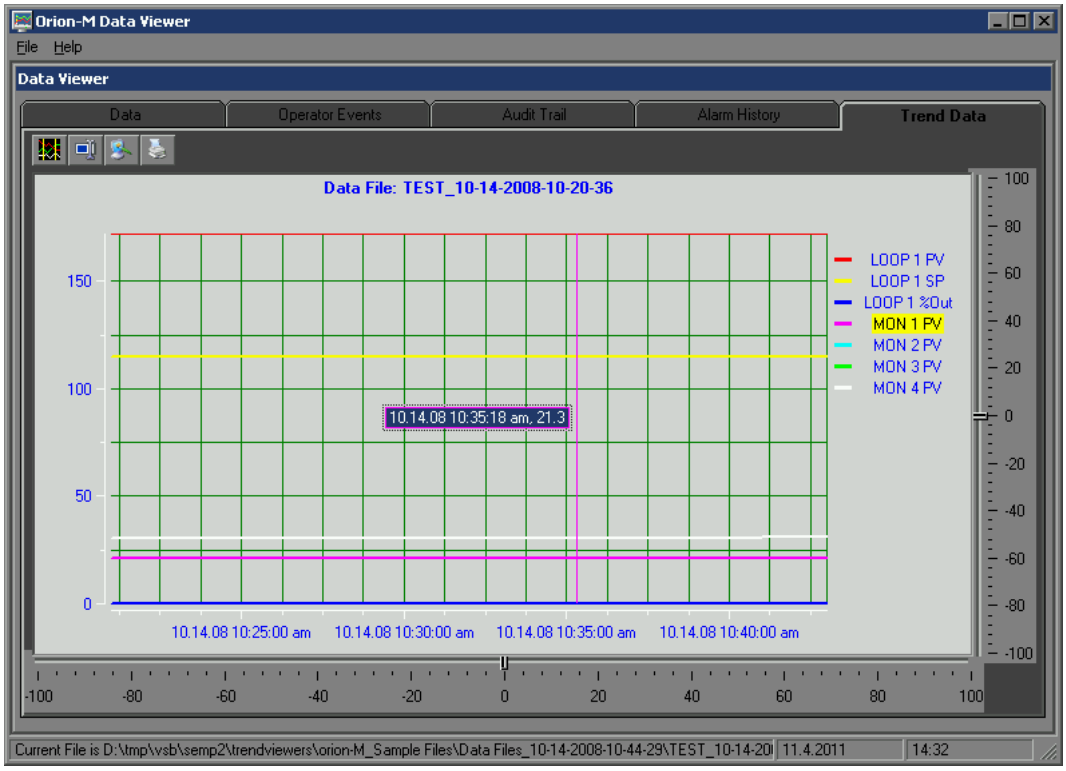
2.3.2 Orion-M Data Viewer

Prohlížeč historický trendů firmy *Future Design Controls* [9]. Jedná se o jednoúčelový prohlížeč určený pro zobrazení dat z jejich vlastních řídicích systémů. Aplikace byla testována s přiloženými daty, která jsou digitálně podepsána a generována právě jejich řídicími systémy. [9]

Jelikož není program univerzální, tak jako ostatní testované nástroje (*TrendAnalyzer* [8] a *ibaAnalyzer* [10]), neoplývá program mnoho funkcemi. Grafické rozhraní aplikace je strohé a je rozděleno do 5 záložek, kde každá záložka má svou funkci:

- „Data“ - zobrazení dat v tabulce.
- „Operator Events“ - seznam zaznamenaných zásahů operátora.
- „Audit Trail“.
- „Alarm History“ - seznam zaznamenaných alarmů.
- „Trend Data“ - zobrazená data v grafu.

Z důvodů zaměření aplikace bylo testování zaměřeno na tu část aplikace, která se zabývá načtením a zobrazením dat do grafu.



Obr. 3: Orion-M Data Viewer [9], testovací data

Data musí být načtena na záložce „Data“. Prostřednictvím otevíracího dialogu je vybrán a následně otevřen soubor CSV (jediný podporovaný typ souborů), se kterým aplikace pracuje, a načten do tabulky. V této tabulce je potřeba vybrat sloupce s daty, které mají být zobrazeny v grafu. [9]

Data jsou zobrazena v jednoduchém grafu (*Obr. 3*). Graf obsahuje pouze jednu osu X a jednu osu Y. Zoom je umožněn prostřednictvím horizontálního a vertikálního posuvníku, které jsou umístěny na dolní a pravé straně grafu. Grafu umožňuje také posouvání rozsahu os X a Y tažením myši, je-li kurzor myši umístěn nad osou. [9]

Nastavení grafu jsou jen ty nejzákladnější – barva jednotlivých křivek a jejich tloušťka. Vložených křivek může být maximálně 12. Mezi další nastavení patří rozsah os X a Y, barvy popředí a pozadí grafu, ale i nastavení kroku mřížky. U osy X je možno nastavit formát času a data, který je zobrazen jako popisek osy. [9]

Důležitou funkcí je možnost využití kurzoru. U zobrazeného kurzoru je možno vidět popisek aktuální hodnoty vybrané křivky, vybraná křivka je zvýrazněna i v legendě grafu. Pomocí kurzorů je možno měřit hodnoty křivek zobrazených v grafu, nebo vymezit začátek či konec křivky. [9]

Zhodnocení

Protože je program určen pro jednoúčelové zobrazování dat, nepotřebuje program mnoho nastavení. Zpracování grafu je velmi jednoduché.

Výhody

- Export dat do CSV souboru.
- Možnost využití kurzoru.

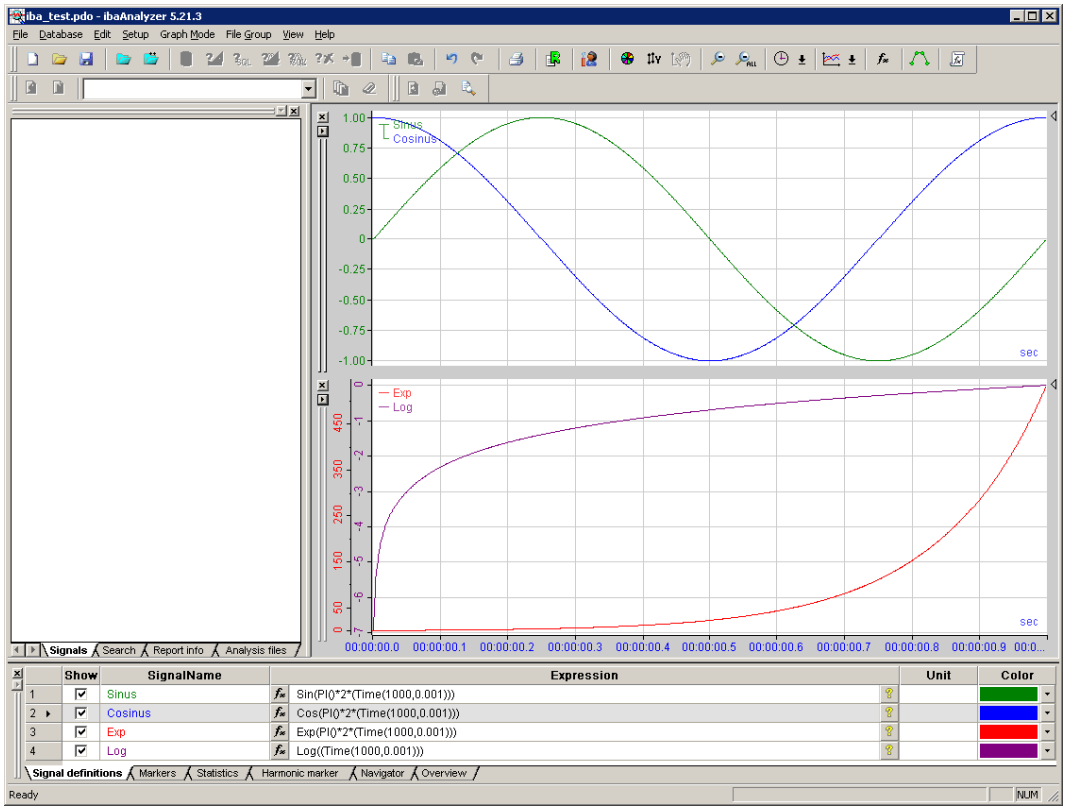
Nevýhody

- Velmi strohé nastavení.
- Nejedná se o univerzální prohlížeč dat.

2.3.3 ibaAnalyzer

Tato aplikace je vyvíjena německou firmou *iba AG*. Hlavním zaměřením firmy jsou měřicí a automatizační systémy. *iba AG* se zabývá jejich vývojem, výrobou a distribucí hardwarových a softwarových komponent právě pro měřicí systémy, systémy sběru a analýzu dat. [10]

ibaAnalyzer [10] je výkonný nástroj pro souhrnnou analýzu dat. Aplikace byla testována ve verzi 5.21.3. Existuje řada různých verzí programu *ibaAnalyzer* [10], každá verze aplikace podporuje různé datové zdroje – práce s databázemi, práce s externími soubory (TXT, CSV, DAT). [10]



Obr. 4: ibaAnalyzer [10], přehledné GUI – seznam signálů, konfigurace křivek, graf

Okno aplikace je rozděleno na 3 hlavní části (*Obr. 4*):

- „Signal Tree“ - strom signálů, seznam signálů dostupných v datových zdrojích.
- „Signal Grid“ - seznam signálů (křivek) zobrazených v grafu.
- „Graph“ - grafické zobrazení signálů.

Okno „Signal Tree“ obsahuje ve výchozím zobrazení seznam signálů datových zdrojů. Jsou zde funkce spojené se signály v datovém zdroji a tyto funkce jsou rozděleny do čtyř záložek. Aplikace umožňuje pracovat s několika datovými zdroji najednou. Hlavní záložka má název „Signals“ a zobrazuje stromový pohled na signály uložené v datových zdrojích, „kořenovou“ položkou stromu jsou reprezentovány jednotlivé datový zdroj. Tyto datové zdroje jsou dále větveny do jednotlivých signálů obsažených v datovém zdroji (jednotlivé položky stromu signálů). Datový zdroj, a strom v něm obsažených signálů, je možno přidat kliknutím pravého tlačítka myši do okna a z nabídky vybrat položku „Add new data file ...“ nebo „Import file tree ...“. V tomto okně jsou obsaženy další záložky. „Search“ je určena pro hledání signálů v datových zdrojích. Další záložkou je „Report info“, ve které je umožněno zobrazit některé charakteristické hodnoty analýzy. Poslední záložka „Analysis files“ je důležitá, protože umožňuje uložení právě prováděné analýzy, tzn. do souboru budou uloženy, pro možné budoucí pokračování v analýze, zobrazené signály, virtuální signály, matematické výrazy, které byly se

signály prováděny, nebo nastavení jednotlivých os grafu. Je plně na uživateli, které signály, matematické výrazy a nastavení má být uloženo. [10]

V okně „Signal Grid“ je obsaženo 5 záložek, první z nich – „Signal definitions“, je určena pro definici signálů a matematických výrazů s nimi spojenými. Obsahuje seznam signálů a virtuálních signálů, které jsou určeny pro zobrazení v grafu. Tyto signály jsou přidávány přetažením vybraného signálu ze stromu signálů nebo je možno je přidat pravým tlačítkem myši pomocí volby „Add signal“. Uživatel má možnost definovat vlastní jméno signálu, zvolit barvu, jednotku a v neposlední řadě vepsat matematický výraz pro operaci s daným signálem. [10]

Přepnutím na záložku „Markers“, jsou v oblasti grafu zobrazeny dva vertikální kurzory či markery. Tyto markery mohou být přemisťovány tažením myši a v okně „Signal Grid“ je možno pozorovat aktuální hodnoty získané z křivek pomocí markerů. V okně jsou zobrazeny i rozdíly, získané z osy X, mezi oběma markery. [10]

Záložka „Statistics“ obsahuje statistické funkce. V grafu zůstanou i po přepnutí na tuto záložku zobrazeny markery, pomocí kterých je vymezena část křivek, která má být využita pro zjištění minima, maxima, střední hodnoty a standardní (směrodatné) odchylky, zjištěné hodnoty jsou vepsány do tabulky k jednotlivým signálům. [10]

„Harmonic marker“ je určena pro zobrazení výsledků FFT analýzy pro hlavní frekvence a jeho harmonické. [10]

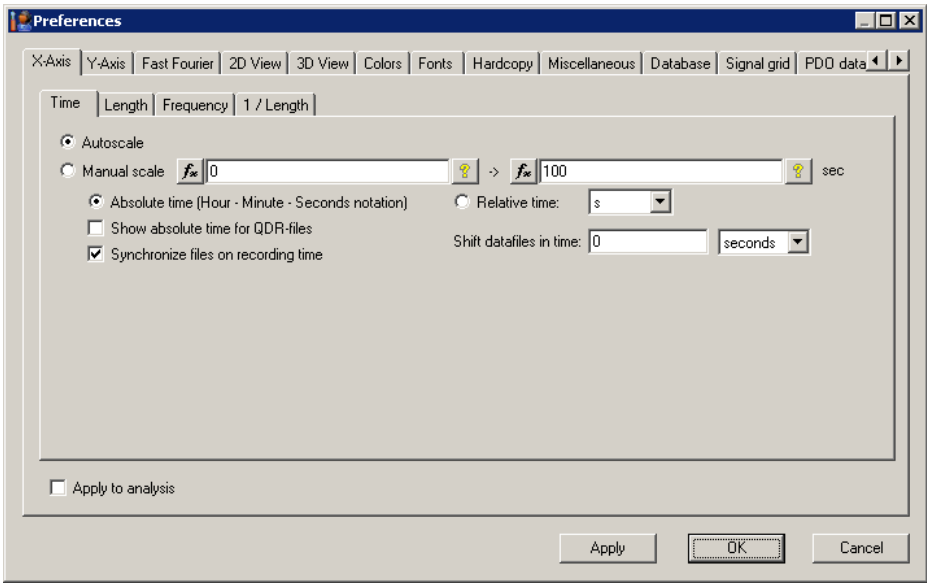
Záložka s názvem „Navigator“ umožňuje zobrazit celkový pohled na graf, ve kterém je vyznačena aktuální pozice v hlavním grafu – tato funkce umožňuje lepší orientaci v grafu. [10]

Přidáním nového signálu do seznamu signálů (záložka „Signal definitions“), je zobrazen v oblasti grafu nový graf určený pro nově vložený signál. Plochu grafu lze velmi dobře konfigurovat, jednoduchým přetažením názvu signálu v legendě pomocí myši, je možné sjednotit všechny signály do jednoho grafu. Dle rozlišení signálů je pak možné dále rozdělit osy Y pro jednotlivé signály. Pomocí kolečka myši a zaměřením osy kurzorem myši je jednoduché přibližovat jednotlivé osy. Posouvání rozsahu os je podobně jednoduché, pouze není využíváno kolečko myši, ale je potřeba stisknout levé tlačítko myši a táhnout směrem, kterým je potřeba posunout rozsah osy. Graf má pro rychlou změnu typu osy X, dostupnou nabídku. Nabídka umožňuje změnit typ os X do několika režimů, čímž je změněna jednotka a typ hodnot osy X: čas (sekundy), frekvence (Hz, režim FFT), délka (metry) a X-Y graf. Mohou být nastaveny také různé režimy zobrazení grafu – klasický 2D pohled, 2D pohled z hora a 3D zobrazení grafu. [10]

Nastavení programu je detailní a propracované. Otevřený dialog pro nastavení má mnoho záložek od nastavení osy X, osy Y, nastavení FFT analýzy, 2D a 3D zobrazení grafu, až po nastavení importu a exportu nastavení (*Obr. 5*). [10]

Pro osu X je možno volit z předdefinovaných veličin – čas, délka, frekvence. U každé z veličin je možno volit automatické nebo manuální nastavení rozsahu osy. Je-li osa X zvolena jako frekvenční, je možno navíc vybrat logaritmické měřítko a nastavení FFT analýzy. Nastavení osy Y obsahuje pouze volbu automatického nebo manuálního nastavení rozsahu osy, a styl zobrazení hodnot osy Y (standardně, vědecky). Uživatel může nastavit 2D a 3D možnosti

pohledu grafu – nastavení průhlednosti, typu grafů pro analogové a digitální signály (čárový graf, pouze body). Záložka „Colors“ je věnována definicím barev okna aplikace a grafům. Další nastavení se zabývá písmem os, legendy, popiskům markerů a písmu v tabulce definic signálů. Zbývající nastavení je věnováno vzhledu stromu signálů, tabulce signálů a nastavení exportu a importu dat. [10]



Obr. 5: Dialog pro s nastavením aplikace ibaAnalyzer

Zajímavou volbou aplikace je automatické přidělení barev signálům, tzn. aplikace, dle algoritmu sama zvolí, které barvy jsou pro signály v grafu nejvhodnější.

Program obsahuje i další pokročilé funkce jako návrhář digitálních filtrů, editor a generátor zpráv o analýze, návrhář maker.

Zhodnocení

Program byl vyzkoušen pouze s ručně vloženými křivkami, protože nebyli dostupné data ve formátu DAT, který program potřebuje. Jelikož je aplikace velmi rozsáhlá, nebyly vyzkoušeny všechny možnosti programu a některé byly popsány pouze teoreticky.

Aplikace má velmi rozsáhlé možnosti. Umožňuje jednoduše přidávat/odebírat signály z datových zdrojů, přidávat virtuální signály, které se definují pomocí matematických výrazů. Editor matematických výrazů a definic virtuálních křivek se velmi podobá matematickému řádku pro zadávání vzorců v *Excelu*, takže je jejich vytváření intuitivní. Obdobně funguje i provádění matematických výrazů mezi signály. Velké množství funkcí je implementováno tak, aby mohla být co nejvíce využívána metoda „Drag & Drop“.

Výhody

- Export analyzovaných dat do souborů DAT, TXT a souboru COMTRADE.
- Široké možnosti nastavení os, analýz, programu.
- Statistické funkce a matematické funkce.
- Navigátor pozice v hlavním grafu, pokud je přiblížena část křivky.
- Návrhář digitálních filtrů.

Nevýhody

- Nebyla nalezena možnost vložit horizontální kurzory (markery).
- Mnoho verzí – může způsobovat nepřehlednost.

2.3.4 Siemens WinCC – komponenta Trend Control

Siemens WinCC [11] je vizualizační software určený pro vizualizaci procesů, tzn. SCADA nebo HMI systém. *WinCC* se nejčastěji používá pro vytvoření vizualizace procesů řízených PLC. Tento softwarový balík obsahuje komponentu *Trend Control*, ta umožňuje zobrazovat jak reálné, tak historické data. [11]

Komponenta *Trend Control* obsahuje základní funkce datových prohlížečů. Mezi tyto funkce patří – přibližování křivek, posun v grafu, možnost zobrazení více os. Při zobrazování reálných dat lze tok dat pozastavit. [11]

Osy lze různě konfigurovat – přidání více os X a Y. Je zde možnost procentuálního vyjádření popisků osy Y. Komponenta má integrovány základní statistické funkce – výpočet minima, maxima, průměru, standardní (směrodatné) odchylky a výpočet integrálu. *Trend Control* umožňuje zobrazit křivku i v logaritmických souřadnicích. Vybrané zobrazené hodnoty lze exportovat do CSV souboru nebo vytisknout. [11]

Zhodnocení

Jde o standardní vizualizační komponentu programu *Siemens WinCC* [10]. Komponentu nebylo možno vyzkoušet, proto byly možnosti popsány dle ukázky ve firmě *Ingeteam*.

Výhody

- Export vybraných dat do CSV souboru.
- Statistické funkce.

Nevýhody

- Nejedná se o samostatnou aplikaci.

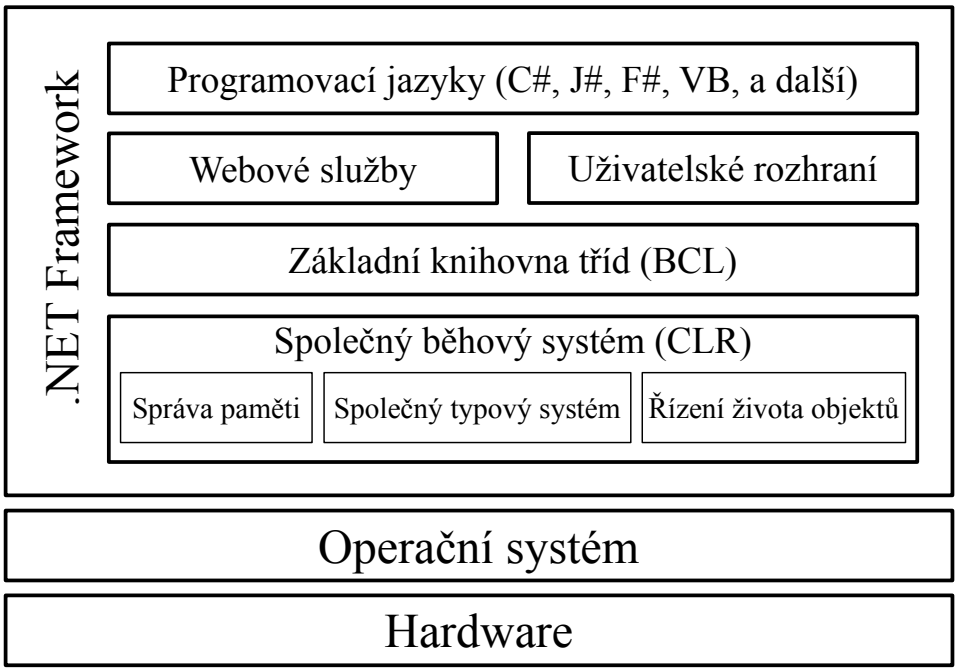
3 Návrh prohlížeče

Kapitola se zabývá zpracováním teoretických možností poskytovaných platformou .NET pro účely vizualizace trendů v průmyslové automatizaci. Inspirace analyzovanými prohlížeči pro návrh vlastního řešení byla důležitá. Tato kapitola se věnuje návrhu prohlížeče, volbou již existujících komponent, které by se daly využít v prohlížeči.

3.1 Platforma .NET

Prostředí .NET framework je soubor softwarových komponent vytvořený firmou Microsoft. První verze prostředí .NET framework 1.0 vyšlo v roce 2000, dnes je uvolněná již verze 4.0 a vyvíjí se verze 4.5. Tato platforma obsahuje prostředí potřebné pro běh .NET aplikací, které obsahuje jak spouštěcí prostředí, tak potřebné knihovny. Pokud je potřeba spouštět aplikace vyvinuté v prostředí .NET framework, je nutné mít tuto platformu nainstalovánou na počítači. Instalační balík je dostupný z webových stránek firmy Microsoft. [7]

Platforma .NET framework obsahuje několik komponent, jejichž struktura je zobrazena na *Obr. 6*. Jádrem celé platformy je tzv. *společné běhové prostředí* (CLR), které zajišťuje běh .NET aplikací (jeho úkolem je např. správa paměti, ošetřování výjimek). Další důležitou komponentou prostředí je *základní knihovna tříd* (BCL), která obsahuje mnoho tříd poskytující různé funkce, např. čtení a zápis do souboru, vykreslování grafických objektů, nebo manipulaci s XML dokumenty. [3]



Obr. 6: Struktura platformy .NET Framework [3]

Z důvodů kompatibility s firmou *Ingeteam* byl vývoj trendového prohlížeče prováděn pro verzi .NET framework 3.5.

Vývoj .NET aplikací

Nejrozšířenějšími programovacími jazyky pro platformu .NET framework jsou jazyky *C#* a *Visual Basic .NET*. Pro vývoj aplikací je dostupné vývojové prostředí vyvíjené přímo firmou *Microsoft – Visual Studio* (pro konkrétní programovací jazyk např. *Visual Studio C#*). Existují, ale i volně dostupné, open-source řešení vývojových prostředí, mezi ně patří např. *SharpDevelop* [12] nebo *MonoDevelop*.

Programovací jazyk C#

Tento jazyk vyvinul *Microsoft* společně s platformou .NET. Jazyk je založen na programovacím jazyku C++ a Java. Cílem tohoto programovacího jazyka je, aby byl jednoduchý, moderní a objektově orientovaný. [7]

3.2 .NET komponenty

Pro usnadnění vývoje trendového prohlížeče bylo potřeba vyhledat a vyzkoušet několik .NET komponent, které jsou již vytvořeny a vyhledat ty nejvhodnější. Vývoj vlastní komponenty s podobnou funkcionalitou, kterou má již vyvinutá .NET komponenta by mohl trvat dlouhou dobu. Proto dobře vyvinutá .NET komponenta může mít pro programátora velkou výhodou, ten pak nemusí vyvíjet vlastní řešení, ale může využít již vytvořenou, případně, je-li komponenta open-source, může být modifikována a upravena tak, aby měla požadovanou funkcionalitu. Je-li některá komponenta vyvíjena určitou dobu, např. několik let, může být dobře propracovaná a mít i kvalitní dokumentaci (např. *ZedGraph*).

Do vyvíjeného trendového prohlížeče byly implementovány některé vybrané .NET komponenty. Každá komponenta má různou funkci a jinak se používá, proto bylo nutné vybrat nejvhodnější komponenty. Komponenty byly zkoušeny a testovány, a na základě jejich testování pomocí vlastních zkušebních aplikací proběhl výběr těchto komponent. Při testování komponent bylo zohledněna snadnost implementace, funkčnost a škála funkcí komponenty.

Všechny dostupné .NET komponenty byly získány jako dynamické DLL knihovny, v této formě lze jednoduše využít jejich metody a třídy v nich obsažené.

Přehled použitých komponent:

- *ZedGraph*. [13]
- *SourceGrid*. [14]
- *DockPanel Suite*. [15]

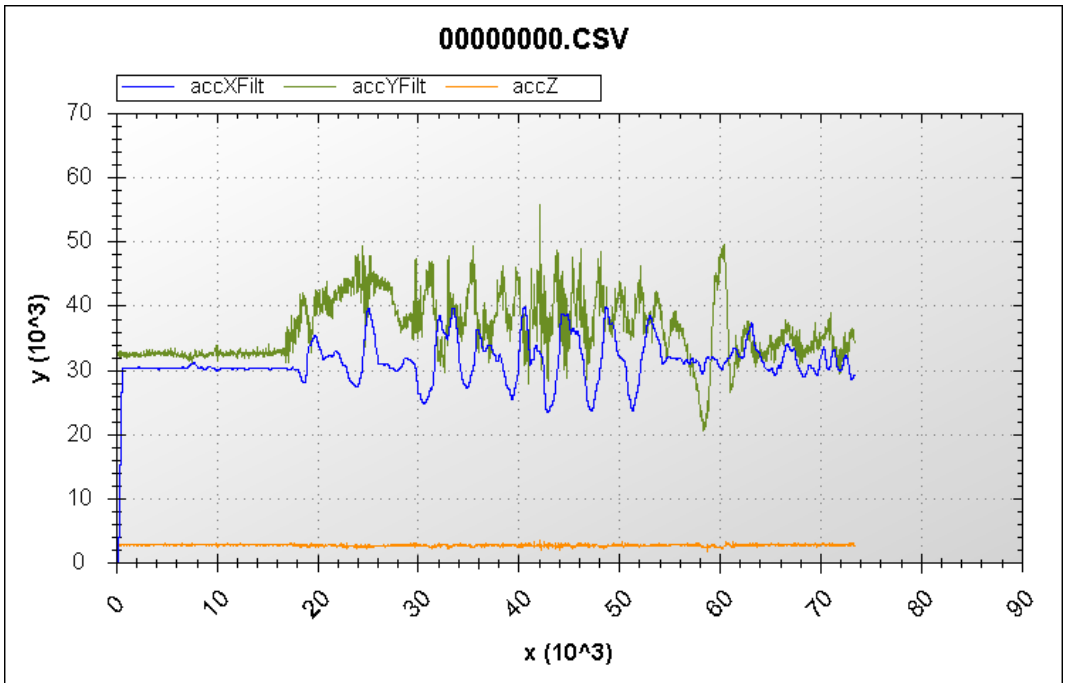
3.2.1 ZedGraph

ZedGraph je open-source knihovna pro platformu .NET framework. Knihovna umožňuje vytvářet jak klasické aplikace (obsahuje grafické uživatelské rozhraní), tak webové

aplikace psané pro ASP .NET (obsahuje webové grafické rozhraní). Jedná se o grafickou komponentu, jak již název napovídá, pro tvorbu grafů – umožňuje vytvářet čárové, sloupcové, koláčové a další 2D grafy. *ZedGraph* je uživatelsky velmi přizpůsobivá komponenta. [13]

Poslední verze komponenty *ZedGraph* byla verze 5.1.5 určená pro .NET framework 2.0. Bohužel již není tato komponenta vyvíjena, ale jelikož jsou platformy .NET framework zpětně kompatibilní, lze tuto komponentu využít i v novějších verzích .NET platformy, např. verze 3.5 nebo 4.0. [13]

Programátor má k dispozici propracovanou nápovědu a na webových stránkách projektu [13] lze naleznout množství příkladů implementace komponenty pro programovací jazyky VB, C# a C++. [13]



Obr. 7: Komponenta ZedGraph s načtenými daty [13]

Grafické uživatelské rozhraní komponenty působí příjemně (*Obr. 7*). Komponenta má implementován jednoduchý zoom pracující s daty zobrazenými v grafu. Mezi další funkce komponenty lze vyjmenovat konfigurovatelnost os grafu, možnost zobrazení popisku křivky s hodnotami křivky při najetí kurzoru myší na křivku, konfigurace rozsahu os (hlavních a vedlejších kroků mřížky), popisky os a grafu, zobrazení/skrytí legendy.

3.2.2 SourceGrid

SourceGrid je open-source komponenta pro .NET framework platformu. Tato komponenta může být použita pro vizualizaci nebo změnu dat ve formě tabulky (*Obr. 8*). Zjednodušeně lze říci, že se jedná o rozšířenou standardní komponentu *DataGridView*, ovšem

právě oproti objektu *DataGridView* umožňuje *SourceGrid* vkládat do jednotlivých buněk tabulky i další grafické objekty – např. tlačítka, zaškrtávací položky či rolovací menu. [14]

Poslední uvolněná verze komponenty byla v roce 2010 verze 4.30. Komponenta obsahuje vlastní grafické rozhraní. *SourceGrid* je napsán v jazyce C# pro .NET framework verze 2.0, komponentu lze samozřejmě využít i v novějších verzích prostředí .NET framework. [14]

	X	Y	SignalName	ValuesCount	Color
1	☑	☐	timeCounter	3626	Blue
2	☐	☐	accXFilt	3626	Blue
3	☐	☐	accYFilt	3626	OliveDrab
4	☐	☐	accZ	3626	DarkOra...
5	☐	☐	trackVoltage	3626	Green
6	☐	☐	motorCurrent	3626	Khaki

Obr. 8: Komponenta SourceGrid, příklad použití [14]

Na webových stránkách projektu [14] lze stáhnout dokumentaci ke komponentě *SourceGrid*, která pomůže programátorovi seznámit se s použitím, přístupem a implementaci komponenty do vlastního projektu.

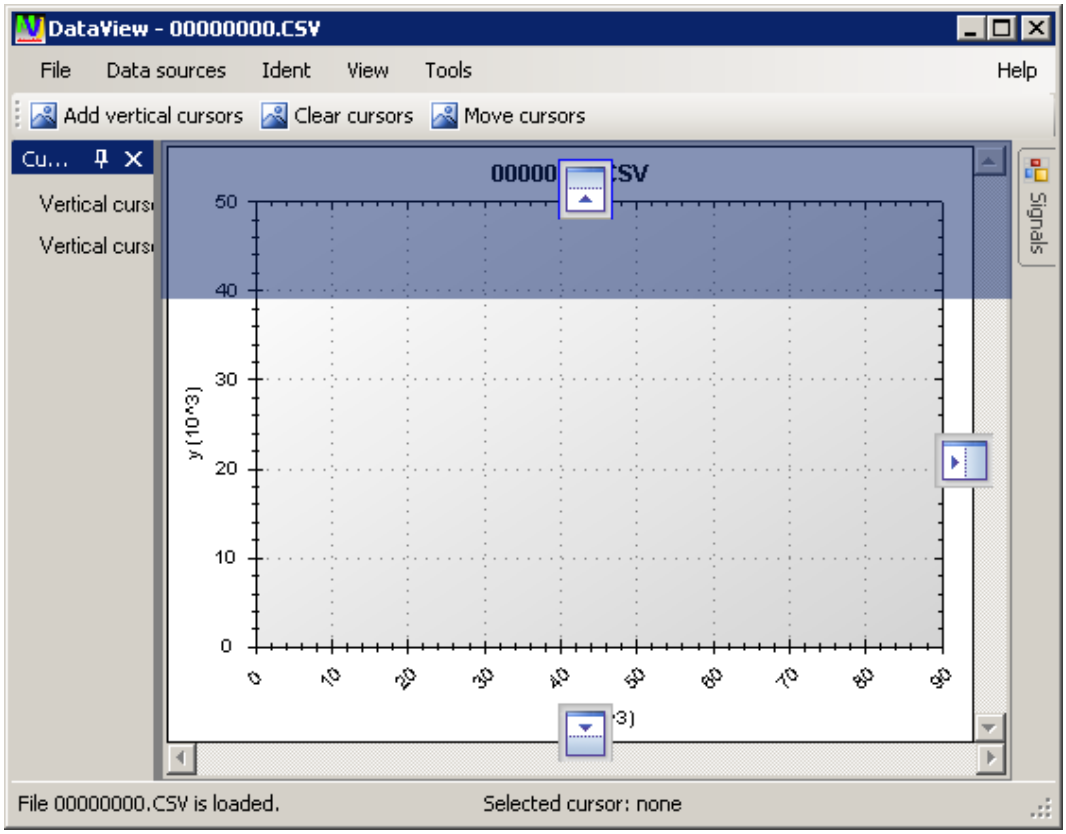
Funkce komponenty jsou velmi rozmanité, do buněk tabulky lze vkládat i další objekty (tlačítka, obrázky, odkazy, apod.), lze měnit pozadí buněk, slučovat více buněk do jedné, nebo přiřadit buňce určitý datový formát či typ (`DateTime`, `Int`, `Boolean`, `String`, `Button`, `Checkbox` a další). [14]

3.2.3 DockPanel Suite

Jedná se o dokovací knihovnu pro .NET framework napodobující vizuální dokovací styl *Visual Studia .NET*. Tato komponenta umožňuje vytvářet plovoucí okna či formuláře, které lze právě pomocí této dokovací knihovny rozmisťovat libovolně po okně v rámci aplikace. Plovoucí okno může být ukotveno k horní, dolní, pravé nebo levé straně aplikace. Další možností je, aby zůstalo plovoucí okno nad oknem aplikace, tzn. nebude neukotveno k žádné straně aplikace, ale bude jakoby „ve vzduchu“ nad aplikací. Pomocí této knihovny mohou být vytvářeny i tzv. MDI aplikace. [15]

V roce 2010 byla vydána verze 2.5. Komponenta *DockPanel Suite* byla uvolněna jako open-source knihovna. Pro programátora je dostupná kvalitní dokumentace a v případě stahování zdrojových kódů komponenty jsou přiloženy i zdrojové kódy vzorového příkladu, což velmi usnadní implementaci komponenty.

Aplikace vytvořená pomocí této knihovny je pro uživatele velmi výhodnou, uživatel si může rozhraní aplikace upravit a přemístit určité plovoucí okna (nebo ukotvit, na libovolnou stranu aplikace) po ploše aplikace tak jak je pro něj důležité.



Obr. 9: Příklad aplikace využívající komponentu DockPanel Suite [15]

Na *Obr. 9* je zobrazeno chování komponenty při přesunu plovoucího okna. Graficky je znázorněno, průhledným obdélníkem, kam bude plovoucí okno umístěno (ukotveno), zatímco malé ikonky znázorňují možnosti, kam je povoleno plovoucí okno umístit. Důležitou funkcí plovoucího okna je možnost skrývání a následné zobrazení při najetí kurzoru myši, což umožňuje úsporu místa v hlavním okně aplikace a nezakrývat tak případné důležité informace.

3.3 Návrh prohlížeče

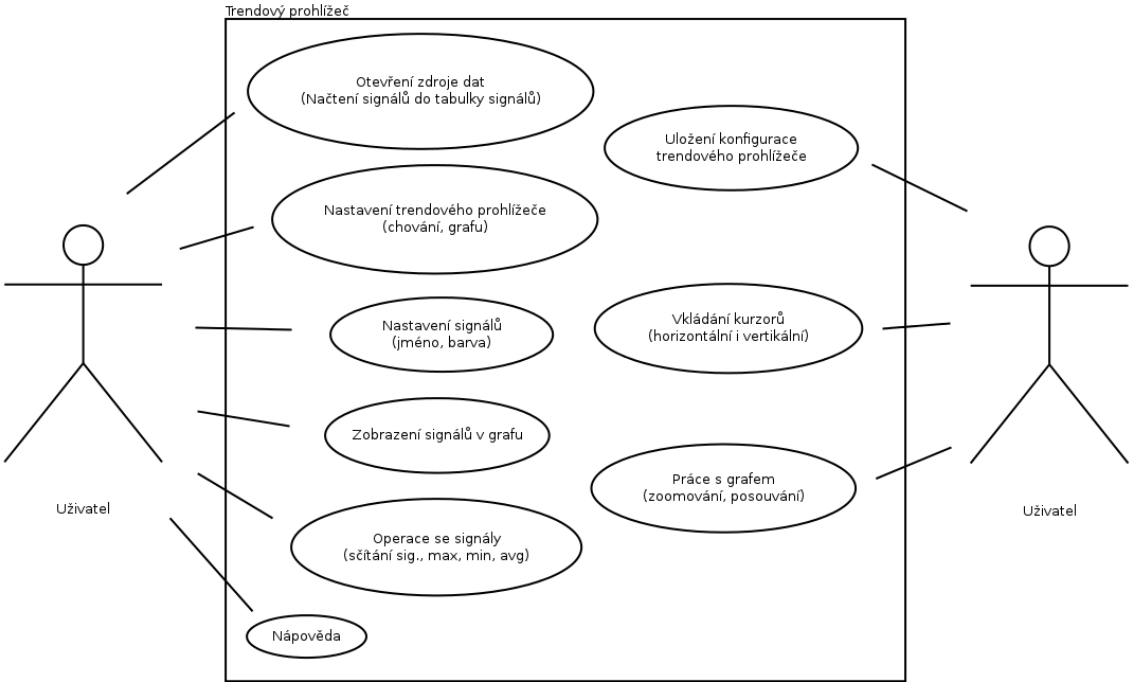
Před návrhem prohlížeče bylo nutné sjednotit a sepsat požadavky. Pokud má být návrh dobře proveden, měli by být známy všechny požadavky a podrobnosti o funkčnosti ještě před samotnou implementací. Budou-li totiž některé požadavky odhaleny až při implementaci, může být problematičtější tyto funkce implementovat.

Návrhem, ale není myšleno jen sjednocení těchto požadavků a zamyšlení se jak vše funguje, ale spadá sem i navržení komunikačního rozhraní s externími knihovnami, nebo návrh grafického rozhraní prohlížeče.

3.3.1 Diagram případu užití

Při několika prvních schůzkách byly zaznamenány požadavky firmy *Ingeteam*. Požadavky byly přepsány do diagramu případu užití. Diagramem byly zachyceny nejen požadavky, ale i vymezeny možnosti trendového prohlížeče (*Obr. 10*, rozšířená verze diagramu případu užití o poznámky a scénáře, viz *Příloha A*). Diagram by tedy měl obsahovat všechny důležité požadavky uvedené v kapitole 2 – požadavky na trendový prohlížeč.

Během implementace jednotlivých funkcí prohlížeče, a následné prezentace změn ve firmě, byly některé požadavky upřesňovány nebo měněny, proto jsou v diagramu zobrazeny pouze obecné požadavky.



Obr. 10: Use-case diagram – komunikace uživatele s prohlížečem

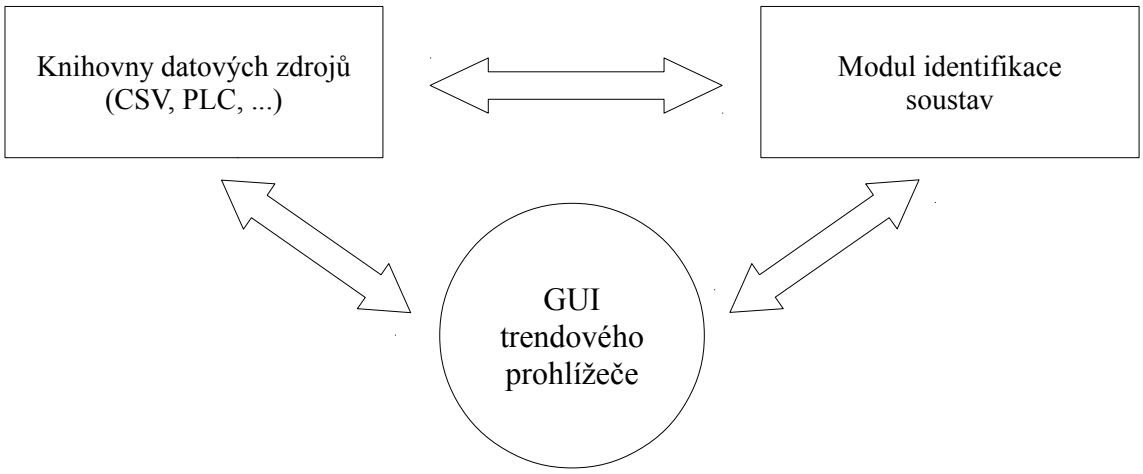
3.3.2 Rozhraní knihoven

Protože GUI je pouze jedna část trendového prohlížeče, a právě touto částí se zabývá tato práce, bylo potřeba vytvořit koncepci komunikačního rozhraní mezi GUI a jednotlivými knihovnami či moduly trendového prohlížeče (*Obr. 11*). Navržením komunikačního rozhraní se rozumí vytvoření obecných předpisů metod, které budou sloužit pro výměnu dat s knihovnami datových zdrojů a modulem identifikace.

Navrhnutí správné koncepce rozhraní bylo pro prohlížeč velmi důležité. Kdyby prohlížeč komunikoval s knihovnami bez rozhraní, pouze přes metody či funkce, bylo by obtížné udržovat prohlížeč stále aktuální, protože by museli být do prohlížeče implementovány stále nové přístupy k jednotlivým datovým zdrojům, což je časově náročné a neefektivní. Tím, že bylo navrženo obecné komunikační rozhraní, je prohlížeči v podstatě jedno s jakým datovým

zdrojem si bude data vyměňovat. Jelikož knihovny pro prohlížeč, a právě v těchto knihovnách je implementováno rozhraní, jsou vyvíjeny dalšími studeny, bylo potřeba komunikovat a dohodnout se, které funkce v rozhraní jsou vhodné, a které naopak ne, jakého datového typu budou jednotlivé návratové hodnoty metod a jakého datového typu mají být argumenty. I proto, že jsou knihovny vyvíjeny někým dalším, bude v následujících odstavcích popsána komunikace pouze z pohledu prohlížeče.

Podrobnější popis, pro přidání a použití knihovny v C# sestavě, bude v následující kapitole 4, která je věnována vlastní implementaci prohlížeče.



Obr. 11: Blokový diagram komunikace mezi GUI prohlížeče a ostatními knihovnami

Komunikace mezi GUI a knihovnami datových zdrojů

Knihovny pro přístup k datovým zdrojům obsahují několik funkcí a událostí, které prohlížeč plně využívá, tzn. ze zdrojového kódu prohlížeče jsou volány jednotlivé metody komunikačního rozhraní. Právě tyto knihovny jsou pro prohlížeč stěžejní, protože jejich prostřednictvím jsou prohlížečem získávána data pro zobrazení křivek.

Seznam metod rozhraní:

```
bool ConfigDataSources();
string[] FindSignals();
int CountValue(int aSigPos);
double[] GetData(int aSigPos, int aStep, int aStartIndex, int
    aEndIndex, bool aCountAreaEnable, bool aSetSampleStep);
string GetDescription();
bool PreviousFile();
bool NextFile();
```

Seznam událostí:

```
event EventHandler Callback;
```

Zavoláním metody `ConfigDataSources()` bude vytvořeno GUI se správcem datových zdrojů. Další metodou je `FindSignals()`, jejíž úkolem je dle nakonfigurovaného správce datových zdrojů předat prohlížeči seznam signálů. Získané pole seznamu signálů nesmí být nijak řazeno, protože prohlížeč je s knihovnami datových zdrojů synchronizován podle indexu jednotlivých signálů v získaném poli. Právě index signálu se dále používán jako vstupní parametr pro další 2 funkce. `CountValue()` získá dle argumentu, což je index signálu, počet hodnot (vzorků) na signál. `GetData()` je metoda, pomocí které jsou získávána data z datových zdrojů pro vykreslování křivek, důležitými argumenty jsou index signálu, vzorkovací krok a počáteční a koncový index, což je využíváno při zoomu. Metodou `GetDescription()` je předáván popis použitého datového zdroje či datových zdrojů, v prohlížeči je možno tuto metodu využít např. pro naplnění nadpisu grafu. Mezi poslední metody implementované v rozhraní patří `PreviousFile()` a `NextFile()`, tyto dvě metody jsou určeny speciálně pro CSV soubory. Pomoci těchto metod, jsou-li použity uživatelem, mohou být procházeny CSV soubory v rámci adresáře, odkud byl CSV soubor otevírán.

Velmi důležitou součástí rozhraní je i událost, pomocí které je umožněno konfiguračnímu oknu datových zdrojů dát vědět prohlížeči, že je vše nakonfigurováno. Na základě této události, může být načten prohlížečem seznam signálů do tabulky signálů.

Komunikace mezi GUI a modulem identifikace soustav

Aby byla možná identifikace soustavy ze zobrazených křivek, bylo nutné vytvořit rozhraní i mezi prohlížečem a knihovnou pro identifikaci. Rozhraní knihovny pro identifikaci obsahuje pouze jednu metodu:

```
void OpenIdent(int aXSignal, int aInSignal, int aOutSignal, int
    aStartIndex, int aEndIndex);
```

Metodou jsou předávána identifikační knihovně informace o datech na ose X, vstupním a výstupním signálu, a pomocí počátečního a koncového indexu je vymezena část signálu pro identifikaci.

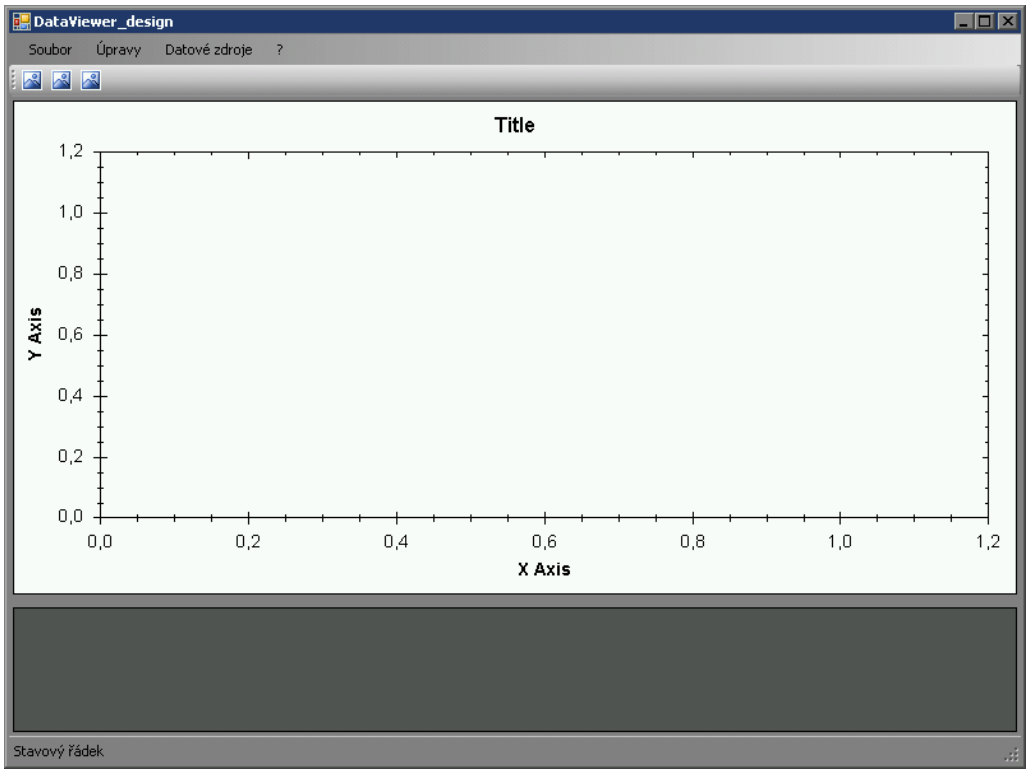
Mezi prohlížečem a identifikačním modulem, nejsou předávány přímo data křivek, ale pouze informace o signálu, protože zobrazená data v prohlížeči mohou být z důvodu rychlosti zobrazování podvzorkována. Proto identifikační modul využívá i komunikaci přímo s knihovnami datových zdrojů, odkud jsou získána pro identifikaci data v plném rozsahu, tzn. v nejjemnějším rozlišení.

3.3.3 Návrh GUI

Jeden z prvních návrhu GUI trendového prohlížeče je zobrazen na *Obr. 12*. Tento návrh byl několikrát přepracován dle požadavků a výsledný vzhled je vidět na *Obr. 13*.

Jednou z největších změn v GUI oproti prvním návrhům bylo použití komponenty

DockPanel Suite. Tím bylo umožněno, aby jednotlivé komponenty nebyly pevně vestavěny v okně aplikace, ale aby byly konfigurovatelné a daly se různě přetahovat ze strany na stranu, a uživatel tak měl plnou kontrolu nad tím, co má být zobrazeno.



Obr. 12: První návrh GUI trendového prohlížeče

V konečné verzi návrhu byly využity 2 dokovací formuláře:

- *Signals* – formulář s tabulkou signálů.
- *Cursors* – formulář s hodnotami získaných z kurzorů.

Dalším zásahem bylo použití více nástrojových lišt než jedné. Může tak být vybráno, které lišty jsou důležité a mají být zobrazeny.

Seznam nástrojových lišt:

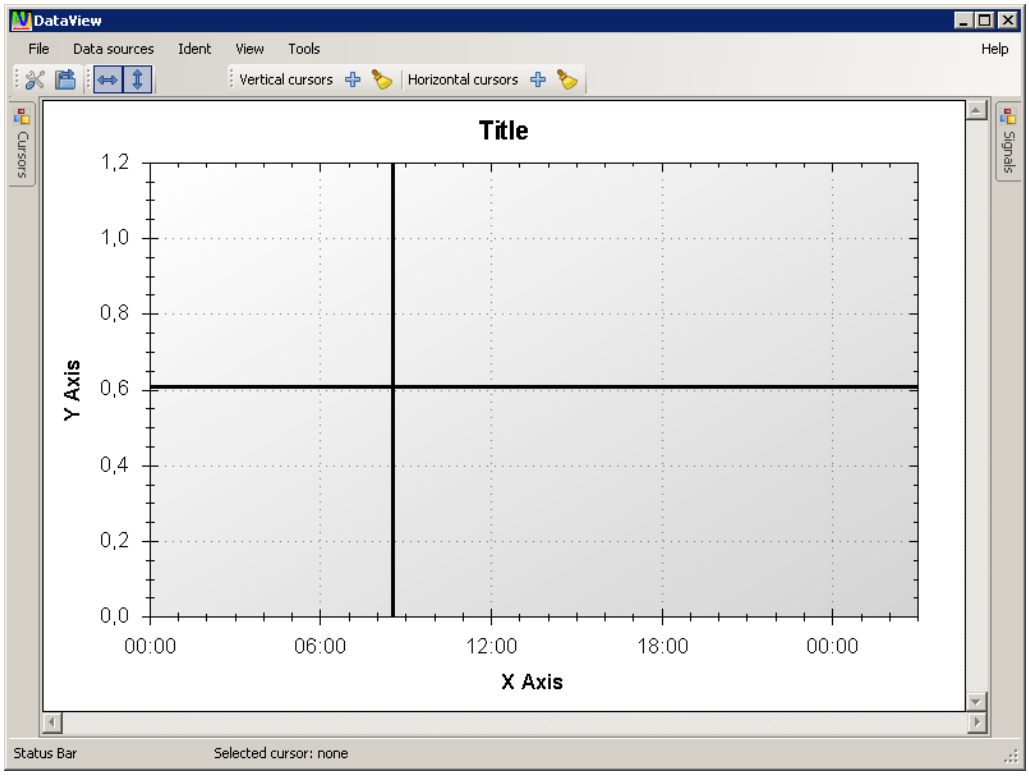
- *Cursors* – umožňuje vkládat a mazat horizontální a vertikální kurzory.
- *Data sources* – pro rychlý přístup k nastavení datových zdrojů.
- *File navigation* – rychlá navigace mezi CSV soubory.
- *Zoom* – povoluje/zakazuje horizontální a vertikální zoom.

S čím bylo počítáno i při prvním návrh byl *stavový řádek*, ten byl pouze rozšířen o další položku, skrz kterou je uživatel informován o tom, který kurzor je právě aktivní.

Samozřejmostí je lišta s nabídkami odkud jsou dostupné některé funkce. Nabídky mají podobnou strukturu, jako většina programů.

Struktura nabídek:

- *File* – otevření a uložení nastavení tabulky signálů a prohlížeče, ukončení aplikace.
- *Data sources* – zobrazení konfiguračního formuláře datových zdrojů.
- *Ident* – zobrazení formuláře pro identifikaci soustav.
- *View* – zobrazení/skrytí nástrojových lišt, dokovacích formulářů a stavového řádku.
- *Tools* – přístup k nastavení aplikace.
- *Help* – nápověda programu.



Obr. 13: Finální podoba GUI prohlížeče

3.3.4 XML pro uložení nastavení

Jedním z požadavků firmy, byla možnost ukládání nastavení do souboru. Později bylo doplněno, aby se jednalo o strukturovaný XML soubor. Právě díky strukturovanosti dokumentu se v něm může dobře orientovat jak člověk, tak program, kterým je dokument zpracováván. Pokud bude správně zvolena struktura, a názvy jednotlivých elementů a argumentů, může být

uživatelem nastavení změněno teoreticky ještě před spuštěním prohlížeče (např. editováním souboru s nastavením v textovém editoru).

XML je obecný značkovací jazyk, vyvinutý a standardizovaný konsorciem *W3C*. Při tvorbě takovéhoto dokumentu je potřeba, aby byla striktně dodržována struktura a syntaxe, na níž je XML závislé. [6] Tento jazyk je určen hlavně pro výměnu dat mezi aplikacemi nebo pro publikování dokumentů. [6]

Proto, aby mohl být XML dokument považován za správně strukturovaný, musí být dodrženy alespoň tyto vlastnosti dokumentu: [6]

- Musí mít právě jeden kořenový element. [6]
- Elementy, které nejsou prázdné musí být ohraničeny startovací a ukončovací značkou, tzv. tagem. [6]
- Elementy, které jsou prázdné mohou být ohraničeny značkou „prázdný element“. [6]
- Hodnoty atributů musí být uzavřeny buď apostrofem (') nebo uvozovkou (“), ale apostrof musí být ukončený apostrofem a uvozovka uvozovkou. Uvnitř hodnot může být použit opačný pár uvozovek. [6]
- Elementy mohou být zanořovány, ale nesmí se překrývat, tzn. každý element (kromě kořenového) musí být celý obsažen v jiném elementu. [6]
- XML dokument rozlišuje velká a malá písmena, tzn. je tzv. case-sensitive. [6]

Příklad jednoduchého XML dokumentu

```
<?xml version="1.0" encoding="utf-8"?>
<ViewerSettings>
    <WindowSize width="800" height="600" />
    <IsShowToolbar>true</IsShowToolbar>
    <ToolbarLocation>
          <X>59</X>
          <Y>0</Y>
    </ToolbarLocation>
</ViewerSettings>
```

První řádek obsahující verzi XML dokumentu a kódování dokumentu je nazýván XML deklarace, u správně strukturovaného dokumentu je nepovinný.

Element `<ViewerSettings>` je tzv. kořenový (root) element, je jím uzavírán celý XML dokument.

Element `<WindowSize width="800" height="600" />` se nazývá prázdný element, protože zde není žádný obsah elementu. `Width` je název atributu elementu a `800` je hodnota atributu.

`<IsShowToolbar>` je počáteční značka (tag) elementu, `</IsShowToolbar>` je ukončovací tag a `true` je obsah elementu.

4 Implementace prohlížeče

Tato kapitola se zabývá samotnou implementací trendového prohlížeče. Pro vývoj byla vybrána platforma .NET framework verze 3.5. Jako vývojové prostředí byl vybrán open-source projekt s názvem *SharpDevelop* [12] ve verzi 3.2. Jedná se o bezplatný ekvivalent k placené verzi *Visual Studia C#* od firmy *Microsoft*. Jeho výhodou je, že je bezplatný a tím, že jde o otevřený software, tak umožňuje i vývoj komerčních aplikací.

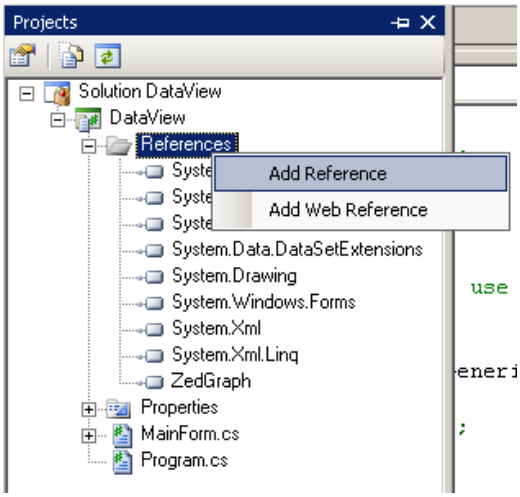
Obr. 14: Logo IDE SharpDevelop

4.1 Reference

Má-li být využívána již vytvořená DLL knihovna (např. knihovna s již vytvořenou komponentou), tedy její metody a třídy, musí být tato knihovna přidána do sestavy. Do sestavy, ve které bude knihovna využívána, musí být přidána tzv. reference na DLL knihovnu.

Přidání reference do sestavy

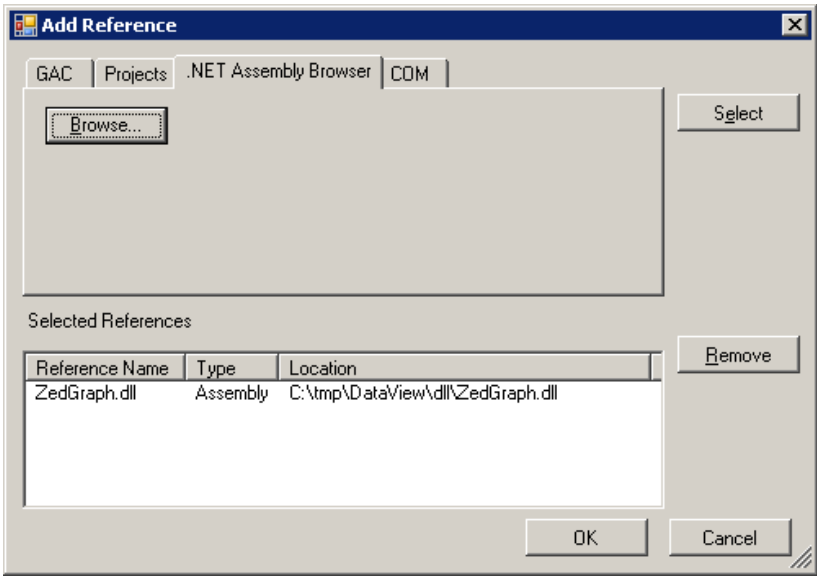
Pokud je potřeba přidat do sestavy referenci na DLL knihovnu, musí být zobrazen strom sestavy, kde se nachází položka „References“. Otevřením položky je možno vidět standardní reference na systémové jmenné prostory přidané při vytváření nové sestavy (*Obr. 15*).



Obr. 15: Reference v sestavě

Přidáním reference na DLL knihovnu je do sestavy přidán tzv. jmenný prostor. Jmenný prostor seskupuje metody a třídy, které spolu souvisí, spadají do podobné kategorie. Například jmenný prostor `System.IO` sjednocuje metody a třídy, pomocí kterých se přistupuje k souborům, adresářům, nebo čtou a zapisují datové toky z/do souborů.

Na *Obr. 15* je zobrazena nabídka pro přidání reference. Pro přidání je potřeba aby byla vybrána první položka „Add Reference“. Výběrem této položky bude zobrazeno dialogové okno (*Obr. 16*). Na záložce „.NET Assembly Browser“ se nachází tlačítko pro vyhledání přidávaných referencí. Zde je již klasicky, prostřednictvím dialogu pro otevírání souborů, potřeba vyhledat cestu k DLL knihovně, která má být přidána. Referenci, která bude následně přidána do sestavy je zobrazena v seznamu zvolených referencí (*Obr. 16*). Referenci lze přidat i přes nabídku „Project“ a položku „Add Reference“.



Obr. 16: Dialog pro přidání reference na DLL knihovnu

Protože byla přidána do sestavy reference na DLL knihovnu *ZedGraph.dll*, přibyla ve stromu sestavy, v adresáři „References“ položka `ZedGraph`. Ihned po přidání reference je automaticky dostupný jmenný prostor z DLL knihovny, který je připraven k používání.

Aby bylo možno jmenný prostor využívat bez zbytečných dlouhých zápisů, který značně znepřehledňuje zdrojový kód, je dobré, aby byl na začátku souboru se zdrojovým kódem definován používaný jmenný prostor použitím klíčového slova `using`. Jmenný prostor, který je využíván v rámci sestavy je definován takto:

```
using ZedGraph;
```

Kdyby nebyl využit tento zápis s klíčovým slovem, byl by přístup k jednotlivým třídám a metodám zdlouhavější, muselo by být použito i jména jmenného prostoru. Např. při vytváření nové instance třídy osy X by musel být použit následující zápis:

```
ZedGraph.XAxis lMyXAxis = new ZedGraph.XAxis();
```

Bude-li ovšem jmenný prostor definován pomocí klíčového slova `using`, zápis bude zkrácen a tvorba instance třídy osy X bude vypadat přehledněji:

```
XAxis lMyXAxis = new XAxis();
```

Obsahuje-li některá z přidávaných knihoven grafickou komponentu, bude tato komponenta automaticky přidána do seznamu komponent a je možno ji využívat v návrháři GUI standardním způsobem.

4.1.1 Použité reference

Do sestavy trendového prohlížeče bylo přidáno několik referencí na knihovny vybraných komponent. Další reference, které bylo nutno přidat, byly na knihovny kolegů – knihovny datových zdrojů a knihovnu pro identifikaci soustav. V následující tabulce je seznam přidaných použitých referencí, respektive jména jmenných prostorů, které obsahují DLL knihovny (*Tab. 1*).

Jmenný prostor	*Popis*
`ZedGraph`	Tvorba 2D grafů – plocha grafu, osy, křivky.
`SourceGrid`	Tvorba tabulek – buňky tabulky.
`WeifenLuo.WinFormsUI.Docking`	Dokování formulářů.
`dataInterface`	Interface pro přístup k datovým zdrojům.
`dataCSV`	Algoritmus čtení z datového zdroje (CSV).
`identifikace_vypocty`	Interface a GUI formuláře pro identifikaci soustav.

Tab. 1: Přehled použitých jmenných prostorů

Pozn. Jmenné prostory obsažené v knihovnách mají většinou stejné jméno jako DLL knihovny, ale bez přípony, proto jsou v tabulce uvedeny jen jmenné prostory.

4.2 Ukládání nastavení, serializace

Pro ukládání nastavení aplikace byla vytvořena vlastní třída `ViewerCfg`. Touto třídou je seskupeno veškeré nastavení aplikace. Uvnitř třídy `ViewerCfg` jsou implementovány další třídy z nichž jsou vytvořeny objekty. Každý objekt této třídy obsahuje určitou část nastavení, čímž je nastavení rozděleno do logických bloků, struktura této třídy je popsána níže.

- Třída `ViewerCfg` – seskupuje nastavení prohlížeče, grafu, kurzorů a os X a Y do jednoho objektu. Z této třídy je vytvořena instance třídy v hlavním okně prohlížeče.
  - Třída `General` – nastavení prohlížeče, tzn. velikost okna, zobrazení jednotlivých oken, zobrazení nástrojových lišt a jejich pozice, šířka křivek a symboly bodů křivek. Z třídy je vytvořen objekt uvnitř třídy `ViewerCfg`.
  - Třída `Graph` – nastavení plochy grafu, tzn. zobrazení a typ nadpisu, font nadpisu, barva plochy grafu, zobrazení a pozice legendy. Z třídy je vytvořen objekt uvnitř třídy `ViewerCfg`.
  - Třída `Cursors` – nastavení kurzorů, tzn. šířka a barva kurzorů. Ve třídě `ViewerCfg` jsou vytvořeny dvě instance – pro nastavení horizontálních a vertikálních kurzorů.
  - Třída `Axis` – nastavení os, tzn. zobrazení a typ popisku osy, font popisku, barva osy, zobrazení osy v grafu, rozsahy osy (minimum a maximum), nastavení hlavní a vedlejší mřížky. Tato třída je společná pro osy X a Y. Je proto vytvořeno 10 objektů, dva pro osy X a 8 pro osy Y, kde je uloženo jejich nastavení.

Aby bylo nastavení prohlížeče uloženo, je potřeba uložit obsah vytvořeného objektu s nastavením do nějakého trvalého úložiště – XML souboru. To, že bude uložena instance do perzistentního úložiště (např. právě do XML souboru), tomu se říká tzv. serializace. Jedná se o to, že objekt je vlastně převeden na proud dat, a proto může být uložen např. právě do souboru. Opakem serializace, a to čtení a opětovné naplnění instance se nazývá deserializace. [3]

Třída, jejíž objekt má být serializován/deserializován musí být deklarován atributem `Serializable`, a musí obsahovat hodnoty základních datových typů, odkazy na objekty, které jsou deklarovány právě tímto atributem, nebo pole základní datových typů. [3]

Třída pro serializaci a deserializaci

Aby mohlo být nastavení prohlížeče uloženo, bylo nutné vytvořit třídu pro serializaci/deserializaci. Třída obsahuje několik metod pomocí kterých je instance serializována. Při ukládání je serializovaný objekt převeden a rozdělen dle typu instance na řetězec, který je pak uložen do souboru.

Protože barva (instance typu `Color`) není deklarován atributem `Serializable`, nastala při serializaci tohoto objektu výjimka. Byly proto do třídy pro serializaci/deserializaci

barvy implementovány právě dvě metody. Jedna metoda převádí barvu na řetězec, aby mohla být uložena, a druhá převádí a rozděluje řetězec na objekt typu `Color`, aby byla instance opět správně načtena.

Nastavení aplikace je ukládáno pod názvem *DataView.xml* do adresáře aplikace, kde je umístěn binární spustitelný soubor programu.

Nastavení je čteno při spouštění aplikace. Ukládání nastavení probíhá během zavírání aplikace nebo po změně nastavení a jeho potvrzení ve formuláři s nastavením.

Uložení nastavení tabulky signálů

Jsou-li používány podobné datové zdroje, ve kterých jsou stejné signály a stejné signály je nutné i zobrazovat, je důležité aby byla ukládána i konfigurace celé tabulky signálů. V signálové tabulce jsou obsaženy důležité informace o tom, které signály jsou zobrazeny, na kterých osách grafu, jakou barvu zobrazené křivky mají nebo jakou mají šířku (*Obr. 17*).

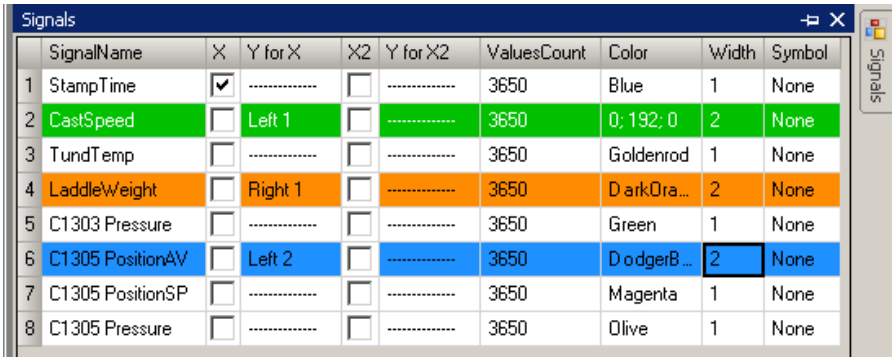
Pro tento účel byla vytvořena nová třída `CurveCfg` určená pro serializaci nastavení prohlížeče a tabulky signálů do jednoho XML souboru. Třída obsahuje instanci třídy `ViewerCfg` s celkovým nastavením prohlížeče a objekt typu `SignalsCfg` s uloženými stavy tabulky.

Před uložením nastavení je potřeba, aby byla naplněna instance třídy `SignalsCfg` tabulkou signálů. Cyklem `for` je tabulka procházena a jsou ukládány jednotlivé řádky. Pak je již snadné celý objekt s nastavením serializovat.

Ukládané nastavení tabulky signálů, je uloženo do adresáře, kde je umístěn datový zdroj. Název XML souboru je převzat ze jména datového zdroje, ke kterému je přidáno „*_DataView.xml*“, aby bylo rozeznatelné, že jde o nastavení prohlížeče s uloženou tabulkou signálů.

4.3 Tabulka signálů

Tabulka signálů je důležitým prvkem prohlížeče. Jsou zde přehledně zobrazeny signály datových zdrojů (*Obr. 17*). Pro vytvoření tabulky byla použita komponenta *SourceGrid* [14].



	SignalName	X	Y for X	X2	Y for X2	ValuesCount	Color	Width	Symbol
1	StampTime	☑	----------	☐	----------	3650	Blue	1	None
2	CastSpeed	☐	Left 1	☐	----------	3650	0; 192; 0	2	None
3	TundTemp	☐	----------	☐	----------	3650	Goldenrod	1	None
4	LaddleWeight	☐	Right 1	☐	----------	3650	DarkOra...	2	None
5	C1303 Pressure	☐	----------	☐	----------	3650	Green	1	None
6	C1305 PositionAV	☐	Left 2	☐	----------	3650	DodgerB...	2	None
7	C1305 PositionSP	☐	----------	☐	----------	3650	Magenta	1	None
8	C1305 Pressure	☐	----------	☐	----------	3650	Olive	1	None

Obr. 17: Tabulka signálů, tři označené signály jsou zobrazeny v grafu

Je-li otevírán uživatelem nový datový zdroj, je tabulka naplněna novými daty – seznamem signálů z otevíraného datového zdroje. Tabulka má vždy 10 sloupců, jejichž význam je následující:

1. Číslování řádků.
2. *SignalName* – jméno signálů.
3. *X* – volba dat na osu X.
4. *Y for X* – volba osy Y pro zobrazení křivky (pro osu X).
5. *X2* – volba dat na osu X2.
6. *Y for X2* – volba osy Y pro zobrazení křivky (pro osu X2).
7. *ValuesCount* – počet bodů (vzorků) na signál (má pouze informativní charakter).
8. *Color* – barva křivky.
9. *Width* – šířka signálu.
10. *Symbol* – symboly bodů křivky.

Aby byl signál zobrazen v grafu, je potřeba, aby byly vybrána data pro osu X nebo X2, podle toho, která osa bude využívána. Výběr dat na osy X byl vyřešen pomocí zatržítek. Bude-li uživatelem požadováno zobrazení signálu aniž by měl vybránu osu X, program takovýto postup uživateli nepovolí.

Sloupce pro výběr os Y, jsou dva z důvodů přiřazení jednotlivým osám X. Výběr os Y byl vyřešen pomocí objektu typu `ComboBox`. Křivku je možno přiřadit jedné z osmi os Y – čtyři osy Y na levé a čtyři osy Y na pravé straně grafu. Je-li osa Y vybrána, signál bude zobrazen jako křivka v grafu a řádek tabulky bude obarven barvou křivky v grafu (*Obr. 17*).

Nastavení signálů může být měněno i když jsou signály zobrazeny v grafu. Pokud je tedy změněna barva, šířka nebo symbol křivky, vše se ihned projeví v grafu. Při změně barvy je změněna i barva pozadí řádku signálu.

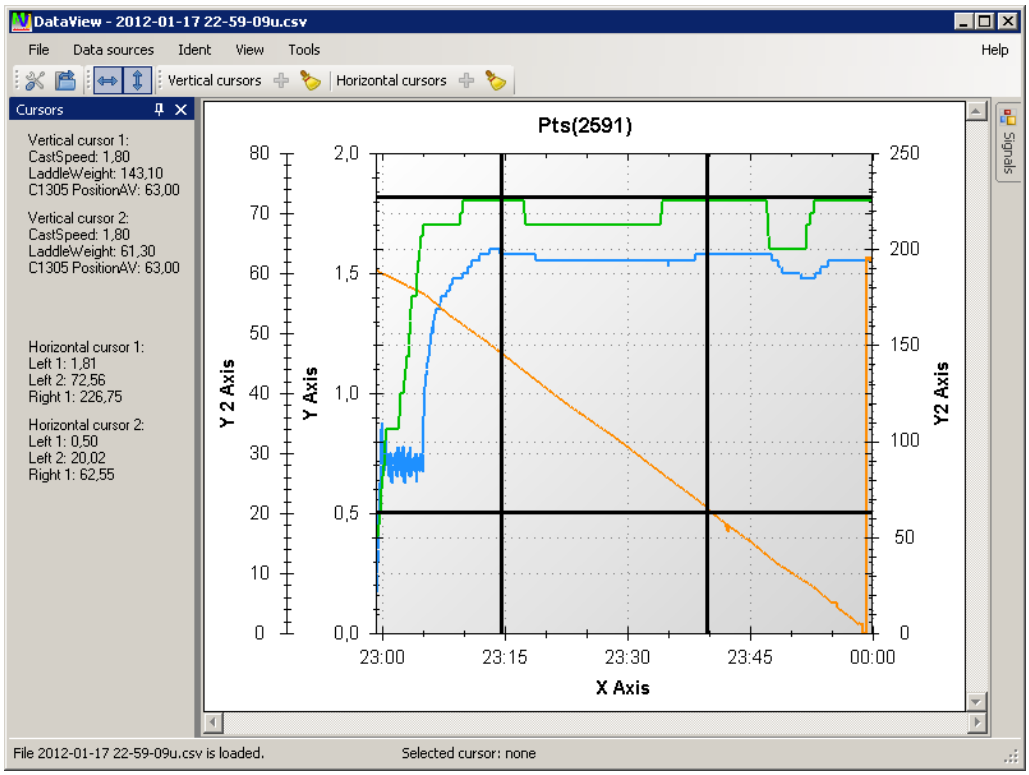
Při změně barvy řádku, když je přidáván signál do grafu, vznikl problém s nečitelností textu řádku, byla-li barva pozadí tmavá. Bylo otestováno, že je-li součet složek barvy (R - červená, G - zelená, B - modrá) menší než 255, je již text nečitelný. Proto při tmavších odstínech je text měněn na bílý.

Pro změnu hodnot buněk tabulky byla využita událost `OnValueChanged`, která je implementována v komponentě *SourceGrid* [14]. Pomocí argumentů události jsou předávány informace o buňce, ve které nastala změna hodnoty, tzn. jsou známy důležité informace jako nová hodnota buňky a souřadnice buňky (číslo řádku a sloupce), kde došlo ke změně. Právě pomocí čísla řádku je identifikováno, u kterého signálu došlo ke změně a pomocí sloupce je filtrováno k jaké změně došlo – zda byla změna osa Y, nebo změna barva, šířka nebo symbol bodů křivky.

4.4 Kurzory

Kurzory jsou nedílnou součástí trendových prohlížečů, měly by usnadnit odečítání hodnot z křivek. Tato vlastnost může být přirovnána k tzv. markerům využívaných pro odečítání hodnot u osciloskopů.

Vyvíjený prohlížeč by měl umožnit uživateli vložit nezávisle na sobě dva horizontální a dva vertikální kurzory.



Obr. 18: Graf s maximálním počtem vložených kurzorů, vlevo hodnoty z křivek

Z požadavků bylo srozumitelné jak mají kurzory fungovat. Zde ovšem vyvstala otázka jakým způsobem by bylo nejlepší, aby byla tato funkce implementována. Bylo potřeba, aby byly otestovány oba níže zmíněné způsoby implementace.

Dva možné způsoby implementace:

1. Kurzor jako křivka v grafu.
2. Kurzor jako nezávislý objekt na grafu.

První způsob implementace kurzorů – slepá cesta

Nejjednodušším způsobem implementace se jevila metoda, že jednotlivé kurzory budou zobrazeny jako jednotlivé křivky grafu. Bylo potřeba, aby byla křivka (kurzor) pojmenována

unikátním jménem, kterým by nemohly být pojmenovány ostatní křivky grafu – signály, a aby nebyl kurzor zobrazován v legendě grafu.

Přidávání kurzorů bylo vyřešeno pomocí události komponenty *ZedGraph* [13] – kliknutí do plochy grafu (`MouseClick`). Ve volání této události bylo zjišťováno zda bylo opravdu kliknuto do plochy grafu. Platila-li tato podmínka, byla vyhledána nejbližší křivka signálu, ze které byly získány informace, kterým osám X a Y je křivka přiřazena a pozice kliknutí myši byla přepočítána na hodnotu osy X. Následně vložený kurzor byl vložen na pozici kliknutí a přiřazen osám X a Y nalezené nejbližší křivky signálu.

Když byl kurzor vložen, uživatel mohl změnit pozici kurzoru. Pro tuto funkci byly využity dvě události komponenty *ZedGraph* [13] – již používané kliknutí do plochy grafu (`MouseClick`) a událost pro stisk klávesy klávesnice (`KeyDown`). V obsluze události `MouseClick` bylo pouze zjištěno, zda je kurzor vložen v grafu, tzn. je-li křivka kurzoru v seznamu křivek. Dále bylo zjišťováno jestli bylo kliknuto na křivku kurzoru s přesností +/- 10 pixelů, když byla tato podmínka splněna, do stavového řádku aplikace byla vypsána informace o vybraném kurzoru. V tomto okamžiku přichází na řadu obsluha události `KeyDown`, aby bylo uživateli umožněno pohybovat kurzorem pomocí definovaných kláves klávesnice v nastavení aplikace.

U tohoto způsobu implementace nebyl možno se vyvarovat několika problémům, i proto musela být využita další možnost vkládání kurzorů.

Prvním problémem bylo, že kurzor byl obsažen v seznamu křivek. To vedlo k tomu, že bylo nutné při každé manipulaci s křivkami signálů filtrovat křivky použité pro kurzory – aby se u nich neprojevovalo nastavení signálových křivek (barva, šířka, symboly).

Druhý problém nastal při vkládání kurzoru, když byla zjišťována poloha nejbližší křivky a bodu na této křivce. Kurzor totiž nebyl vložen na souřadnice, které byly zvoleny uživatelem kliknutím myši, ale na nejbližší bod nalezené křivky.

Dalším problémem byla úplná závislost na komponentě *ZedGraph* [13]. To se projevovalo hlavně tím, že nebylo možno kurzory svobodně pohybovat a vkládat je. Při pohybu kurzoru pomoci kláves byl kurzor závislý na použitých datech (hodnotách) na ose X.

Některé z výše zmíněných problémů jsou vyřešeny dalším způsobem implementace kurzorů.

Druhý způsob implementace kurzorů – správná cesta

Do formuláře s grafem byly vloženy čtyři objekty typu `Panel`, tyto objekty pak představují jednotlivé kurzory. Objekty byly přizpůsobeny tak, aby připomínaly kurzory, tzn. u vertikálních kurzorů byla upravena šířka na 3 pixely a výška na rozsah osy Y, a u horizontálních kurzorů byla upravena výška na 3 pixely a šířka na rozsah osy X. Ve výchozím stavu byly kurzory nastaveny jako neviditelné.

Potřebuje-li uživatel využít kurzorů, ať už horizontálních nebo vertikálních, bylo jejich ovládání seskupeno na nástrojovou lištu (*Obr. 19*), pomocí které mohou být do grafu vkládány

a nebo odstraňovány. Lišta (*Obr. 19*) se ovládá pomocí tlačítek, tlačítkem s ikonou „plus“ jsou postupně přidávány do oblasti grafu kurzory, maximálně mohou být přidány 2 horizontální a 2 vertikální kurzory. Zatímco tlačítkem s ikonou „koštěte“ jsou kurzory z oblasti grafu odebrány.

Obr. 19: Nástrojová lišta pro ovládání kurzorů

Ve skutečnosti vlastně nejsou kurzory přidávány, protože jsou pouze neviditelné. Tím, že jsou uživatelem „přidány“ kurzory do grafu, je kurzor pouze zviditelněn. Při odebrání jsou kurzory opět zneviditelněny.

Tím, že jsou kurzory vytvořeny jako nezávislý objekt typu `Panel`, nejsou vázány v okně s grafem pomocí indexů dat osy X, jak tomu bylo v předchozím případě u křivek, ale svou polohou vůči formuláři.

Aby bylo možné s vloženým kurzorem hýbat, bylo nutné použít 3 událostí objektu typu `Panel`:

- `MouseDown` – událost nastane, když je kurzor myši nad objektem a je drženo stisknuté jakékoliv tlačítko myši.
- `MouseUp` – událost nastane, když je kurzor myši nad objektem a je stisknuté tlačítko myši uvolněno.
- `MouseMove` – událost nastane, když se hýbe kurzorem myši nad instancí.

Událost `MouseDown` je využívána proto, aby byla indikována aktivita kurzoru. Hýbat kurzorem je možno pouze v tom případě, je-li kurzor myši nad objektem, je stisknuto levé tlačítko myši a zároveň je kurzorem myši pohybováno (událost `MouseMove`). Je tedy využito kombinace událostí `MouseDown`, `MouseUp` a `MouseMove`.

V události `MouseDown` je pouze naplněna pomocná proměnná typu `bool` hodnotou `true`, a do stavového řádku je vypsáno, který kurzor je aktivní. Událost `MouseUp` je velmi podobná, ale pracuje naopak, tzn. pomocná proměnná typu `bool` je naplněna hodnotou `false`, a informace o kurzoru je ze stavového řádku vymazána. Událost `MouseMove`, je využívána pro přemisťování kurzoru v oblasti grafu, ale zároveň musí být stisknuto levé tlačítko myši, což je indikováno hodnotou `true` pomocné proměnné nastavené v události `MouseDown`. Jsou-li splněny tyto podmínky, je zjištěna pozice kurzoru. Pomocí této pozice je dále zjištěno zda se kurzor nachází uvnitř oblasti grafu. Pohybem kurzoru je přenastavena jeho pozice, velikost a jsou zobrazeny hodnoty z křivek. Je-li kurzorem překročena oblast grafu, je dle poslední platné pozice kurzor nastaven na začátek nebo konec osy X (nebo Y).

Komponenta *ZedGraph* [13] obsahuje událost `Resize`, pomocí které je zjišťováno, kdy byly změněny rozměry grafu. Nastane-li tato událost, jsou měněny pozice a velikosti kurzorů,

které jsou přizpůsobeny právě novým rozměrům oblasti grafu.

Při získávání hodnot křivek, je přepočítána pozice kurzoru z pixelů na body v grafu (pro přepočet byla použita metoda komponenty *ZedGraph*, `ReverseTransform`). Dále je zjištěno, zda je v seznamu křivek alespoň jedna křivka a má smysl dále pokračovat. V dalším kroku je porovnávána přepočtená pozice kurzoru s daty osy X, čímž získáme index na data kde byl kurzor umístěn. Posledním krokem je procházení seznamu křivek, získání hodnot pomoci nalezeného indexu, a zobrazení těchto hodnot.

4.5 Nastavení aplikace

Některé nastavení aplikace je dostupné přes hlavní nabídku. Jedná se hlavně o zobrazení nástrojových lišt, dokovacích formulářů a stavového řádku.

Nastavení jednotlivých částí grafu je četnější, proto byl vytvořen pro sjednocení tohoto nastavení formulář (obrázky formuláře, viz *Příloha C*).

Nastavení je předáno formuláři pomocí konstruktoru, kde jsou jako argumenty požadovány jednotlivé instance s nastavením – objekty s hlavním nastavením, nastavením grafu, jednotlivých os a kurzorů.

Je-li nastavení potvrzeno tlačítkem *OK*, je nastavení převzato zpět z formuláře, uloženo do globálního objektu s nastavením a načteno aby se změny projevily. Nakonec je nové nastavení uloženo do XML souboru.

Formulář je rozdělen do pěti záložek:

- *General* – hlavní nastavení.
- *Graph* – nastavení grafu.
- *X Axis* – nastavení os X.
- *Y Axis* – nastavení os Y.
- *Cursors* – nastavení kurzorů.

Na záložce *General* je obsaženo hlavní nastavení – šířka signálů, symbol bodů signálů a uživatelské nastavení, zda má být použito nastavení vzorkovacího kroku a omezení počtů bodu křivky z nastavovacího GUI datových zdrojů.

Záložka s nastavením grafu je nazvána – *Graph*. Obsahuje nastavení nadpisu grafu – typ nadpisu (automatický – popis použitého datového zdroje, manuální – nadpis je vložen uživatelem, nebo bez nadpisu), dále barvy pozadí a popředí nadpisu a font. Je zde možnost povolit zobrazení legendy grafu a její umístění. Posledním nastavením na této záložce je nastavení barev pozadí oblasti grafu a oblasti kolem grafu.

Nastavování os X a Y je identické, protože je pro ukládání tohoto nastavení použit stejný typ objektu, záložky se liší pouze popisem. Je možno nastavit popis os, toto nastavení je shodné s nastavením nadpisu grafu, pouze zde chybí možnost automatický popis osy. Důležitou volbou je typ osy, zda se jedná o lineární, časovou, logaritmickou nebo řadovou (ordinální) osu. Dalším podstatným nastavením je minimum a maximum osy. Poslední možností je konfigurace

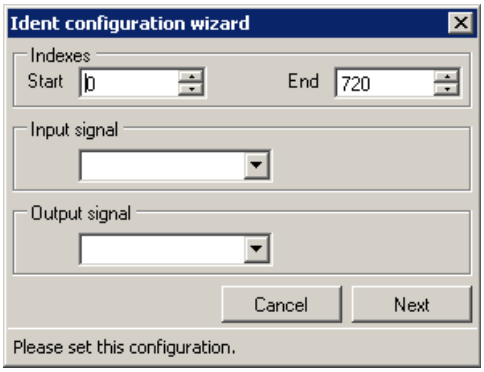
mřížky, má-li být zobrazena, s jakým krokem a barvou.

Nastavení kurzorů je rozděleno na horizontální a vertikální. Zde je možno měnit pouze barvu a šířku kurzoru.

4.6 Výběr signálů pro identifikaci

Prohlížečem je využíván modul pro identifikaci soustavy ze zobrazených křivek ve formě DLL knihovny. Aby byl modul schopen správně identifikovat, je potřeba předat knihovně určité informace, což bylo definováno při návrhu komunikačního rozhraní.

Pro správnou identifikaci je potřeba vstupního a výstupního signálu. Byl vytvořen formulář (*Obr. 20*), pomocí kterého uživatel vybere rozsah signálů (počáteční a koncový index) a dle jména křivek má možnost vybrat vstupní a výstupní signál (identifikačnímu formuláři se předávají pouze indexy signálů). Jsou-li všechny tyto informace vyplněny, je potřeba potvrdit výběr signálů a jejich rozsahů tlačítkem *Next*, tímto je nastavovací formulář ukončen a otevírán formulář pro identifikaci (tento formulář je již součástí knihovny). Otevíranému formuláři jsou předány nastavené informace při vytváření identifikačního formuláře, tedy konstruktorem.



Obr. 20: Výběr signálů pro identifikaci

Proběhlo-li vše v pořádku, je otevřeno okno pro identifikaci. Aby byla identifikace správná, jsou data zvolených signálu předávána přímo z knihoven datových zdrojů, protože v grafu nemusí být zobrazeny všechny vzorky signálu.

4.7 Zoom

Zoom je nedílnou součástí každého prohlížeče. V komponentě *ZedGraph* je obsažen základní zoom pracující s křivkami zobrazenými v grafu, tzn. *ZedGraph* jen překresluje již vložené křivky v grafu. Tento typ zoomu je ale vhodný pouze pokud má křivka maximálně kolem 5000 vzorků. Bylo vyzkoušeno, že je-li vloženo v grafu 5 až 10 křivek a každá má 100000 vzorků, je překreslování již neúnosné a může trvat kolem 5 až 10 sekund. Bylo potřeba vymyslet vlastní algoritmus, který by vyřešil zoom křivek s takovým množstvím vzorků.

Na *Obr. 21* je zobrazena nástrojová lišta pro povolení/zakázání horizontálního a vertikálního zoomu.

Obr. 21: Povolení horizontálního a vertikálního zoomu

Algoritmus zoomování

Použité události komponenty *ZedGraph*:

- `ZoomEvent` – událost nastane při zoomu grafu.

V obsluze události `ZoomEvent` je nejdříve kontrolováno, zda je vytvořen objekt pro komunikaci s datovými zdroji. Dále je zjišťováno, je-li datový zdroj připraven k použití, tedy zda již byl načten seznam signálů do tabulky v prohlížeči. V tomto okamžiku je volána metoda `SignalSampling()`, jejíž vstupními argumenty jsou minima a maxima osy X, respektive osy X2.

V první řadě je kontrolováno, je-li v seznamu křivek nějaká křivka (jestli je nějaká křivka zobrazena v grafu). Pokud by totiž, žádná křivka v seznamu nebyla, metoda by byla ukončena. Jsou uloženy, indexy signálů, jejichž data jsou na obou osách X. Musí být zkontrolováno jestli jsou opravdu nějaká data na ose X či ose X2, je-li vlastně některá z os využita. Důležitým krokem je porovnávání počtu vzorků křivky s rozlišením obrazovky, na základě výsledku porovnávání (je-li počet vzorků křivky větší než rozlišení obrazovky) bude využit zoomovací algoritmus. Zde přichází na řadu metoda pro zjištění indexu `GetIndex()`.

Zjištěné indexy (počáteční a koncový) jsou uloženy a dále budou použity pro získání nových dat z knihovny datových zdrojů. Ještě předtím je, ale porovnáván rozdíl koncového a počátečního indexu s rozlišením obrazovky, následně je vypočítán vzorkovací krok, který je nutné vložit do rozhraní pro získání nových dat. Vzorkovací krok je počítán tak, že je podělen rozdíl koncového a počátečního indexu rozlišením obrazovky.

Pak už jsou jen získána nová data z knihoven datových zdrojů prostřednictvím rozhraní. Nejdříve jsou získány data osy X a pak je procházen seznam křivek pomocí cyklu `for`. Každá křivka, ta reprezentuje jeden signál datového zdroje, je aktualizována zvlášť.

Získání indexu

Jako argument funkce pro získání indexu `GetIndex()` je předávána hodnota minima nebo maxima osy X pro zoomování, obecně je tato hodnota nazvána mezí. Tyto obě meze je potřeba získat, aby mohly být načteny nové data křivek z knihoven datových zdrojů. Ještě než je zavolána tato metoda je při každé změně kopírováno pole hodnot signálu, který byl zvolen uživatelem na osu X.

Jak tedy tato metoda funguje. Hodnota předána argumentem metody, mez grafu, je

přepočtena na procenta, jelikož je známo z uloženého pole dat osy X počet vzorků. Protože je index i ukazatel na vzorek pole stejný, je přepočteno procento na index, čímž je získán ne přesný, ale přibližný index hodnoty meze jejíž index je potřeba zjistit. V této fázi je porovnávána mez s hodnotou získanou z pole dat osy X pomocí vypočteného indexu. Je-li index menší, je vždy o jednotku zvýšen a znovu porovnáván dokud není získaná hodnota vyšší než porovnávaná. Obdobné je řešení, pokud je získaná hodnota pole vyšší než porovnávaná. Pak je index vždy o jednotku zmenšen a porovnáván dokud není hodnota menší než porovnávaná. Nakonec je nalezený index vrácen metodou `GetIndex()` jako návratová hodnota.

4.7.1 Možnosti jednotlivých os

Aby byly možnosti prohlížeče co nejvíce přizpůsobeny uživateli, byly implementovány některé užitečné funkce, které by mohly uživateli usnadnit práci s osami (je jedno zda se jedná o jakoukoliv osu X nebo Y):

- Zoom pouze jedné vybrané osy.
- Posouvání rozsahu pouze jedné vybrané osy.
- Kopírování nastavení jedné vybrané osy na druhou zvolenou osu.

Tyto funkce jsou ovládány pomocí klávesových zkratek v kombinaci s myší. Byly využity tyto události komponenty *ZedGraph* [13]:

- Událost `KeyDown` – nastane při stisku klávesy klávesnice.
- Událost `KeyUp` – nastane při uvolnění stisknuté klávesy klávesnice.
- Událost `MouseDownEvent` – nastane při stisku kteréhokoliv tlačítka myši.
- Událost `MouseWheel` – nastane při rolování kolečkem myši.

Zoom vybrané osy

Přibližování a oddalování vybrané osy pomocí klávesové zkratky <CTRL> + rolováním kolečka myši. Při použití této kombinace, ale musí být zároveň kurzor myši umístěn nad osou, která má být využívána.

Pro implementaci této funkce bylo využito třech událostí – `KeyDown`, `KeyUp` a `MouseWheel`. Pomocí událostí `KeyDown` a `KeyUp` je identifikováno zda je stisknuta nebo uvolněna klávesa <CTRL>. V události `KeyDown` je naplněna pomocná proměnná (typu `bool`) hodnotou `true`, což znamená, že byla klávesa stisknuta, v obsluze události `KeyUp` je tato proměnná naplněna hodnotou `false`, tedy, že byla klávesa uvolněna.

V obsluze události `MouseWheel` je nejdříve využíváno pozice kurzoru myši. Dle této pozice je nalezen nejbližší objekt pod kurzorem myši. Je-li nalezený objekt nenulový a zároveň typu `Axis`, tedy osa X nebo Y grafu, je pokračováno dále ve vykonávání události, v opačném případě je vykonávání metody ukončeno. Zde je využívána pomocná proměnná, pro indikaci

stisknuté klávesy <CTRL>. Pokud je tedy klávesa stisknuta, je osa přibližována (nebo oddalována, dle směru rotace kolečka myši). Pokud, ale není klávesa stisknuta, je metoda ukončena a zoom osy není proveden.

Posouvání vybrané osy

Při posouvání osy vlastně dochází ke změně rozsahu osy, rozsah osy je posouván dle rolování kolečka myši o pevně stanovený krok směrem nahoru nebo dolů. Posouvání rozsahu je dostupné použitím klávesové zkratky <SHIFT> + rolováním kolečka myši. Opět musí být kurzor myši umístěn nad osou, u které má být rozsah posouván.

Zde je využito tří stejných událostí, jako v předchozím případě – `KeyDown`, `KeyUp` a `MouseWheel`. Obslužné metody událostí `KeyDown` a `KeyUp`, jsou využívány naprosto stejným způsobem jako u zoomu osy, s tím rozdílem, že je kontrolována klávesa <SHIFT>, zda je stisknuta, nebo uvolněna.

Uvnitř obslužné metody události `MouseWheel` je nejdříve zjištěna poloha kurzoru, následně je nalezen nejbližší objekt. Aby nebyla událost ukončena, musí být hledaný objekt typu `Axis` (osa X, nebo Y). Pak je zjištěno je-li stisknuta klávesa <SHIFT> a až nyní může být u osy měněn rozsah.

Kopírování nastavení osy na jinou osu

Funkce byla přidána pro rychlou možnost změny rozsahu osy, dle rozsahu jiné osy grafu. Jde tedy o funkci kopírování nastavení osy aniž by musel být otevřen formulář s nastavením, kde by se musel rozsah osy změnit manuálně, což je relativně zdlouhavá činnost.

Funkčnost je tedy následující, kurzor myši musí být umístěn nad osou, ze které bude nastavení kopírováno a zároveň musí být stisknutá klávesa <ALT> (klávesa musí být stisknuta po celou dobu procesu kopírování nastavení), pak stačí kliknout myší na osu a nastavení bude uloženo. Dalším krokem je umístění kurzoru myši nad osu, kam má být nastavení zkopírováno, a kliknutím myši bude nastavení zkopírováno na osu pod kurzorem myši, až nyní může být klávesa <ALT> uvolněna.

Z hlediska implementace této funkce bylo použito tří událostí – `KeyDown`, `KeyUp` a `MouseDownEvent`. Událostmi `KeyDown` a `KeyUp` je indikován stav klávesy <ALT>, zda je klávesa stisknuta nebo uvolněna, pro indikaci je využívána pomocná proměnná. Zároveň je v události `KeyUp` nulován objekt typu `Axis`, kam je ukládáno kopírované nastavení os, a proměnná indikující že, bylo nastavení zkopírováno.

Při kliknutí levým tlačítkem myši na osu je vyvolána událost `MouseDownEvent`. Uvnitř události, je zjišťováno, dle pozice kurzoru myši, zda se kurzor myši nachází nad osou grafu nebo ne. V dalším kroku je zjištěno, jestli je stisknuta klávesa <ALT>, dále je nastavení osy uloženo a nastavena pomocná proměnná indikující uložené nastavení osy na hodnotu `true`. Kopírování nastavení osy probíhá ve stejném sledu – je zjišťováno, zda nejbližší objekt pod kurzorem myši je osa grafu, dále je-li stále stisknuta klávesa <ALT>, až nyní může být nastavení zkopírováno na osu nad níž se nachází kurzor myši. V posledním kroku je nastavena

pomocná proměnná indikující kopírování nastavení na hodnotu `false`.

4.8 Statistické a matematické funkce

Statistické funkce signálu jsou základní informace získané ze signálu – minimum, maximum a průměr (střední hodnota) signálu.

Matematickými operacemi se signálem jsou myšleny základní matematické operace jako sčítání, odečítání, násobení a dělení. Matematické operace by měli fungovat jak se dvěma signály, tak se signálem a libovolným číslem, tzn. například součet dvou signálů nebo součet signálu a libovolného čísla.

Po konzultaci ve firmě zatím není jasné, kde budou tyto funkce implementovány, zda přímo do knihoven datového zdroje nebo do prohlížeče. Proto následuje zamyšlení se nad problémem a jeho rozbor.

Statistické funkce a matematické operace implementované v prohlížeči

Protože v prohlížeči nemusí být zobrazovány všechny data, mohou nastat 2 možnosti:

1) Použít pro výpočty křivky (počet vzorků může být ale podvzorkován), které jsou zobrazeny v grafu.
2) Použít pro výpočty všechny vzorky signálu, které poskytnou knihovny datových zdrojů a zkopírují se do proměnné v prohlížeči.

První případ může zanést do výpočtu chybu. Protože pro zobrazování nemusí být použity všechny vzorky signálu. Může tedy chybět některá důležitá část signálu, v tomto případě by uživatel mohl zjistit zkreslené informace – např. nepravdivé minimum, maximum nebo při součtu signálů by mohla být výsledná křivka zkreslená.

Druhý případ by řešil ztrátu informací, tzn. uživatel by zjistil skutečné informace o signálu. Toto řešení, se ale může zdát docela neefektivní, protože by muselo být zkopírováno do proměnné celé pole vzorků signálu (získané z knihoven datových zdrojů) u kterého jsou zjišťovány statistické funkce, nebo použity matematické operace. I když v dnešní době, kdy se operační paměti počítačů pohybují kolem jednotek až desítek gigabytů, by to nemusel být takový problém. Dalším možným problémem by mohlo být zpomalení celého prohlížeče při výpočtech, protože by se muselo procházet celé pole vzorků signálu.

Možným řešením by bylo použít ke statistickým funkcím a matematickým operacím pouze viditelnou část křivky v grafu, a vyžádat si od knihoven pouze všechny vzorky zobrazené části signálu.

Statistické funkce a matematické operace implementovány v knihovnách datových zdrojů

Možné zefektivnění statistických funkcí by byla jejich implementace do knihoven datových zdrojů. Například při použití s CSV souborem, jsou knihovnou načítány všechny data, tzn. jsou procházeny všechny data uložená v souboru. Již během načítání souboru by mohlo být zjišťováno minimum, maximum a počítána střední hodnota. Tím, že by byly tyto informace

zjišťovány už během načítání, bylo by potřeba do rozhraní knihoven přidat další metody, kterými by byly prohlížeči předávány výsledky statistických funkcí.

U jiných datových zdrojů by to bylo obdobné. Např. u SQL databáze není potřeba načítat celou databázi, ale pouze se dotazovat a získávat ty data, které jsou požadovány prohlížečem. Operace by tedy byly prováděny pouze z toho rozsahu dat či vzorků, jejichž rozsah je zobrazen v grafu.

U implementace matematických operací není jednoznačné, zda by byla implementace do knihoven datových zdrojů výhodou, protože při těchto operacích je nutné procházet pole vzorků signálu a provést operaci pro každý vzorek zvlášť. Jedinou výhodou je snad jen to, že by se nemuselo kopírovat celé pole vzorků do proměnné v prohlížeči, ale pouze by se získalo pole výsledného signálu.

4.8.1 Implementace statistických a matematických funkcí – testovací aplikace

Pro otestování implementace obou funkcionalit byla vytvořena vzorová aplikace.

V testovací aplikaci byly implementovány tyto statistické funkce:

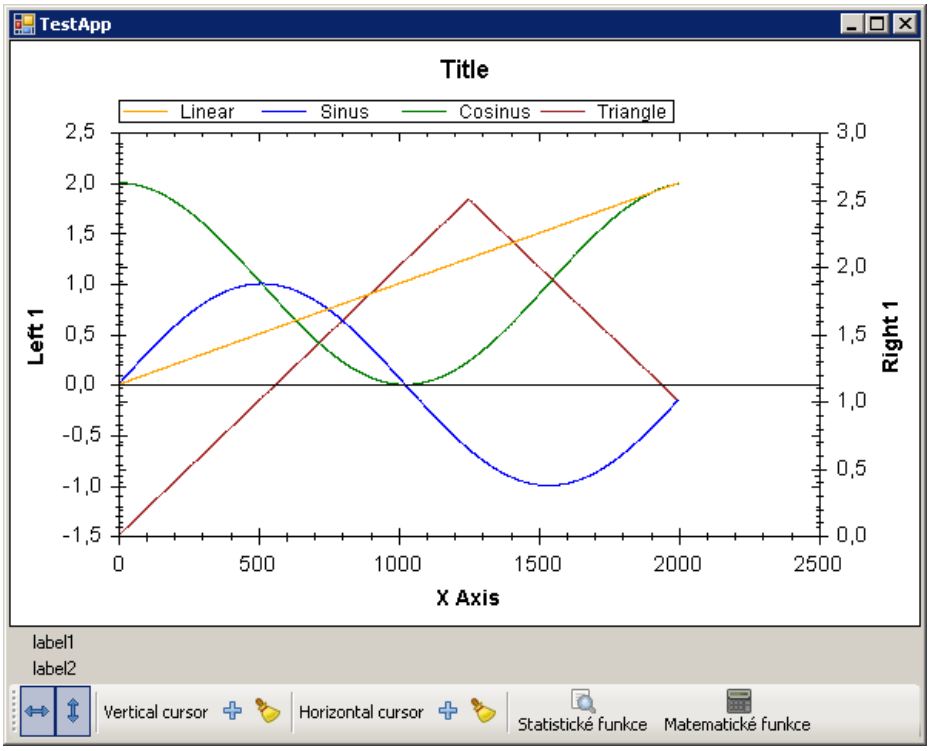
- Minimum.
- Maximum.
- Střední hodnota.

Matematické operace implementovány v testovací aplikaci:

- Sčítání.
- Odečítání.
- Násobení.
- Dělení.

Matematické operace mohou být prováděny jak se dvěma signály, tak s jedním signálem a zvoleným číslem.

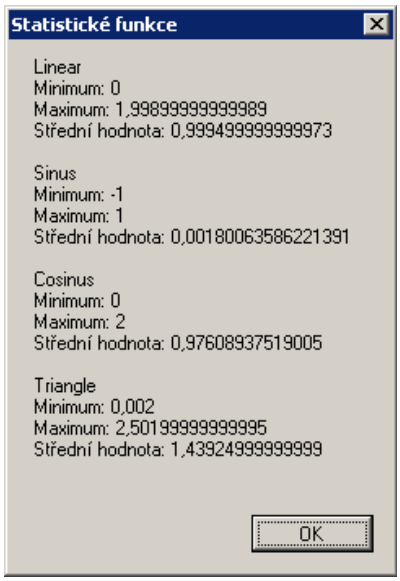
V aplikaci (*Obr. 22*) jsou pevně při spouštění vygenerovány 4 křivky – lineární rostoucí, sinus, kosinus a trojúhelník. Každá z těchto křivek má 2000 vzorků.



Obr. 22: Testovací aplikace pro testování statistických funkcí a matematických operací

Implementace statistických funkcí

Všechny statistické funkce uvedené výše jsou zjišťovány a zobrazovány po stisku tlačítka „Statistické funkce“, kdy je otevřeno dialogové okno s informacemi zjištěnými z křivek (*Obr. 23*).

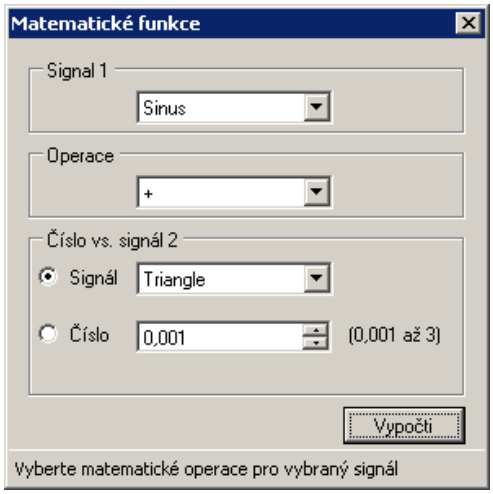


Obr. 23: Dialogový formulář se statistickými funkcemi

Všechny statistické informace jsou zjišťovány obdobně. Byly vytvořeny 3 proměnné typu `double`, pro uložení minima, maxima a střední hodnoty křivky. Aby byly zjištěny informace ze všech křivek, muselo být použito dvou cyklů `for`. Jedním cyklem `for` byl procházen seznam křivek grafu a druhý cyklus `for` byl použit pro procházení pole vzorků.

Zjišťování minima a maxima je implementováno velmi podobně, při inicializaci proměnných byly naplněny prvními hodnotami z pole dat procházené křivky. Při procházení pole byly tyto hodnoty porovnávány se vzorkem na aktuální pozici. Při zjišťování minima, byla-li tedy hodnota aktuálního prvku menší než hodnota uložená, byla uložena hodnota z aktuální procházené pozice. U zjišťování maxima, byla uložena zase hodnota větší.

Při zjišťování střední hodnoty byly všechny hodnoty vzorků sčítány a následně po skončení procházení pole vzorků poděleny počtem vzorků.

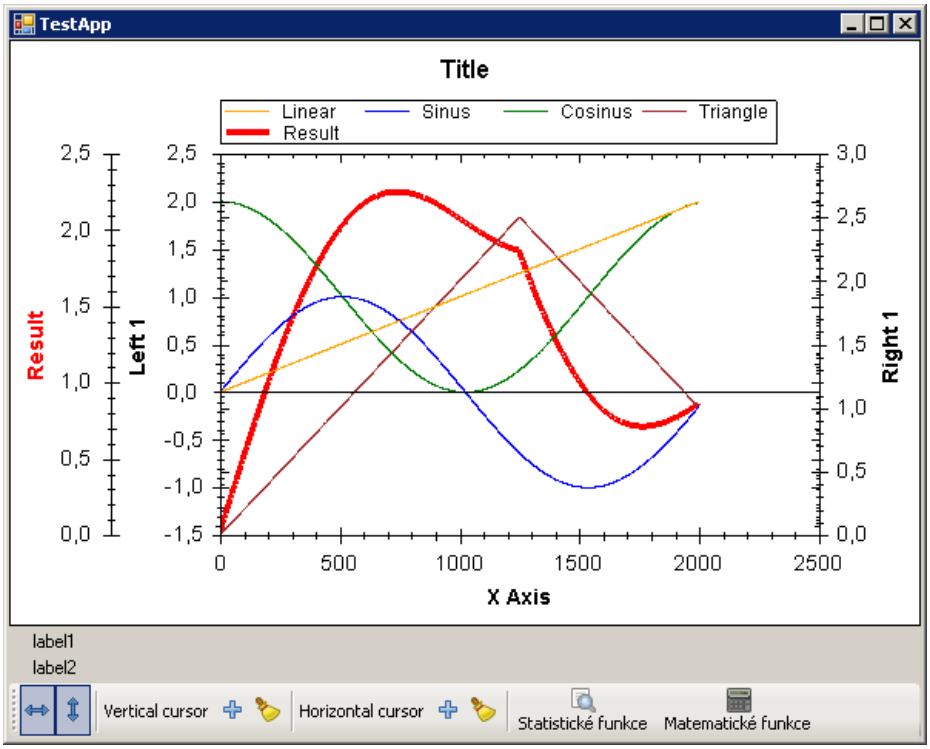


Obr. 24: Formulář volbu matematické operace

Na *Obr. 24* je zobrazen nemodální dialog pro volbu matematických operací. Aby bylo možné operace provádět, je potřeba aby byla zvolena první křivka (signál), dále operace, která bude prováděna a druhá křivka nebo číslo.

Bude-li operace prováděna se dvěmi křivkami, musí být procházeno pole obou křivek najednou a operace je pak prováděna s jednotlivými vzorky. Pokud bylo, jako druhý prvek operace zvoleno číslo, je procházeno pouze pole první křivky a každá hodnota vzorku křivky je s tímto číslem sečtena, odečtena, vynásobena nebo podělena.

Tlačítkem „Vypočti“ (*Obr. 24*) je výpočet proveden a výsledná křivka je zobrazena v grafu na své vlastní ose Y (*Obr. 25*).



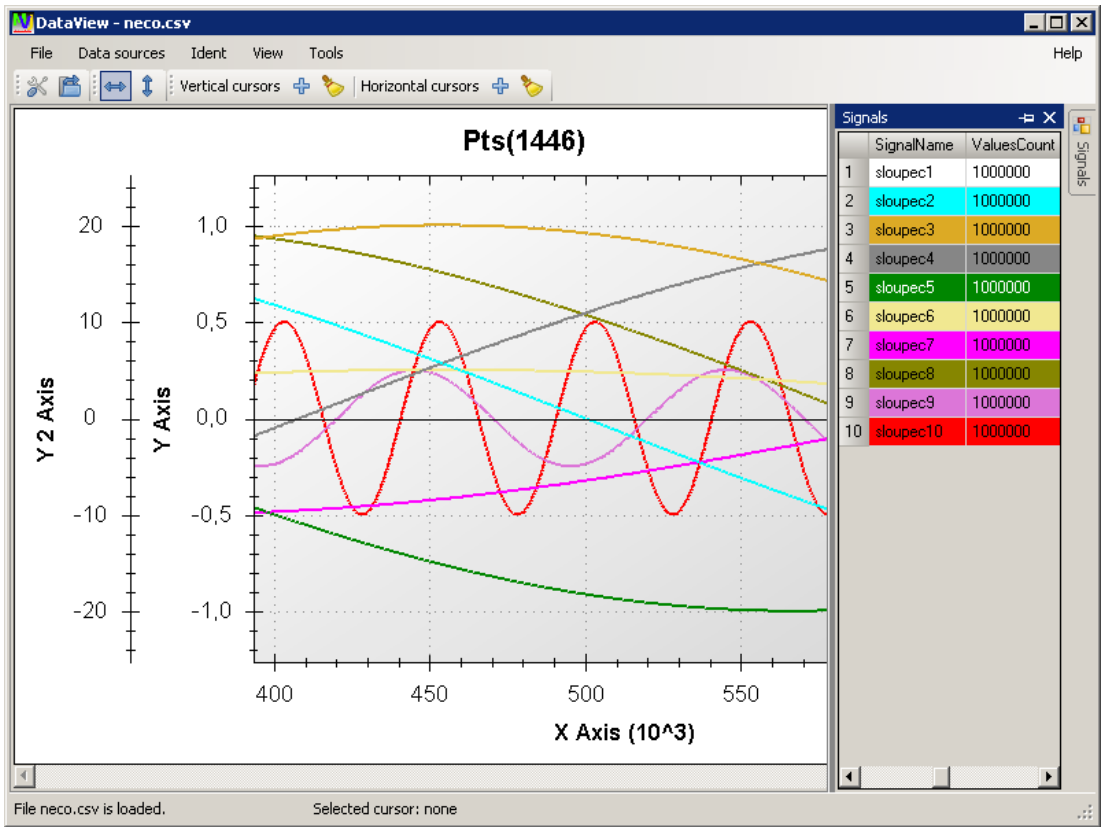
Obr. 25: Testovací aplikace se zobrazeným výsledkem, součet sinu a trojúhelníkového průběhu

5 Stanovení limitů realizované aplikace

Byla vytvořena pomocná aplikace pro vygenerování testovacích dat. Vyvinutý prohlížeč byl těmito testovacími daty otestován. Data byla uložena do CSV souboru.

Parametry vygenerovaných dat:

- Soubor obsahoval 10 signálů:
  - První signál byl určen jako data na osu X.
  - Zbývajících 9 signálů bylo určeno pro zobrazení do grafu prohlížeče.
- Všechny vygenerované signály byly:
  - Sinusového průběhu, různě posunuté.
  - Každý signál měl jeden miliónu vzorků.
- Velikost souboru byla 89,6 MB.

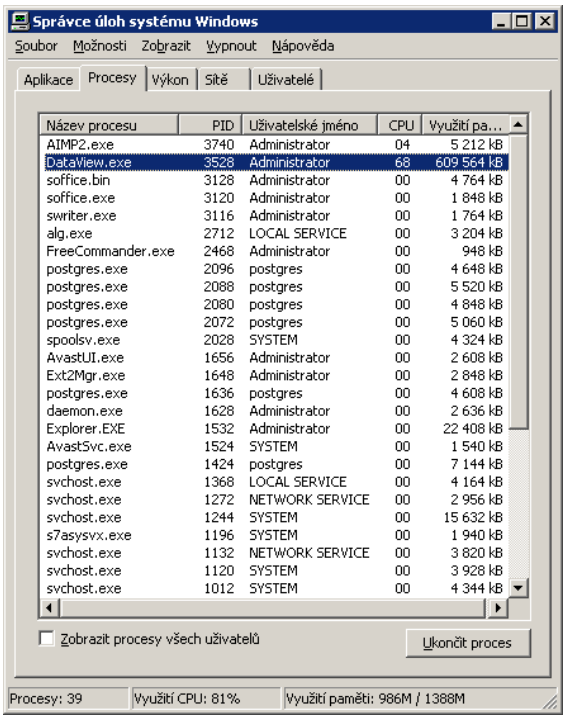


Obr. 26: Prohlížeč se zobrazenými vygenerovanými daty – zobrazeny jsou všechny signály

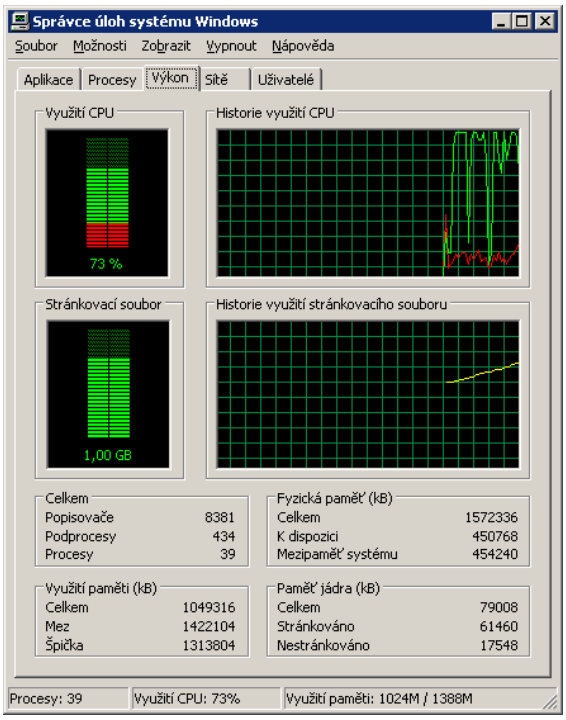
S těmito vzorovými signály byla testována rychlost zoomování, jak se projeví změna šířky zobrazených křivek, změna barva křivek nebo změna symbolu bodů křivek (*Obr. 26*).

Při zoomování testovacích dat bylo zajímavé sledovat jak rostlo využití operační paměti. Byly zjištěny následující údaje:

- Po načtení a zobrazení všech testovacích křivek se využití operační paměti vytvořenou aplikací ustálilo kolem 450 MB.
- Při zoomovaní vzrostlo využití paměti aplikace až k 600 MB (*Obr. 27*).



Obr. 27: Využití CPU a paměti prohlížeče



Obr. 28: Grafy využití CPU a paměti

Na *Obr. 28* je zobrazeno vytížení procesoru a operační paměti při zoomování. Při testování byly na PC spuštěny standardní aplikace – antivirus, kancelářský balík, poslech hudby, a celkové využití operační paměti přesáhlo i 1 GB (*Obr. 28*).

Použitý vývojový a testovací počítač:

- AMD Sempron 2500+, pracovní frekvence 1750 MHz.
- Operační paměť 1,5 GB.
- Operační systém Windows XP Professional SP2.

Vzhledem k tomu, že byl vývoj aplikace a všechny testy prováděny na počítači starém asi 10 let, měla by být aplikace použitelná na všech dnešních počítačích, omezení by nemělo být ani ze strany CPU nebo operační paměti.

5.1 Limity aplikace

Některé limity aplikace byly stanoveny již při návrhu, například počet os a kurzorů. Další možnosti prohlížeče musely být otestovány, například možnosti zobrazování signálů o velkém počtu vzorků – test zoomovacího algoritmu.

Omezení pro zobrazení křivek v grafu:

- Pevně stanovený počet os:
  - Maximálně 2 osy X.
  - Maximálně 8 os Y:
    - 4 osy Y na levé straně grafu.
    - 4 osy Y na pravé straně grafu.
- Pevně stanovený počet kurzorů:
  - Maximálně 2 vertikální kurzory.
  - Maximálně 2 horizontální kurzory.
- Bylo otestováno, že:
  - Prohlížeč umožňuje zobrazovat i křivky, které mají až jeden milión vzorků.
  - Tabulka signálů byla otestována s CSV souborem, který obsahoval až 255 signálů.

6 Závěr

Cílem této diplomové práce bylo vytvoření funkční aplikace – trendového prohlížeče, pro zadávající firmu *Ingeteam*. Vytvoření takového prohlížeče bylo rozděleno do několika bodů.

Prvním krokem byla analýza trhu trendových prohlížečů. Tím bylo zjištěno jak jednotlivé testované aplikace pracují, a jaké mají možnosti.

Dalším bodem práce byl návrh aplikace. Konzultacemi s firmou *Ingeteam*, byly vymezeny požadavky, které byly zaznamenány do diagramu využití. Diagram užití představoval pouze nadhled na celý prohlížeč, jelikož se některé požadavky upřesňovaly až během implementace. V tomto bodu bylo jedním z nejdůležitějších kroků vytvoření komunikačního rozhraní mezi GUI prohlížeče a jednotlivými knihovnami, moduly, které vyvíjeli kolegové. Do návrhu aplikace bylo zařazeno vyhledání a otestování použitelných komponent – DLL knihoven s komponentami pro usnadnění práce při vývoji prohlížeče. Byly vybrány a použity tři knihovny – *ZedGraph* (grafem pro zobrazení signálů), *SourceGrid* (tabulka signálů) a *DockPanel Suite* (dokovací komponenta – rozvržení oken v rámci aplikace).

Během implementace bylo nutné řešit mnoho problémů. Bylo potřeba umět implementovat zvolené komponenty, k čemuž velmi pomohla dokumentace jednotlivých knihoven. Při implementaci vznikalo mnoho otázek, jak by měly být jednotlivé funkce prohlížeče použitelné. Tyto otázky byly konzultovány a řešeny se zadavateli. U některých otázek pomohla inspirace analyzovanými trendovými aplikacemi. Bylo vyřešeno ukládání nastavení aplikace do XML souboru pomocí serializace. Seznam signálů z datových zdrojů bylo potřeba načíst do tabulky signálů. Během implementace kurzorů vznikla časová prodleva z důvodů špatné volby způsobu implementace. Bylo zkoušeno implementovat kurzory jako křivky grafu, ale to se později projevilo jako špatná cesta – filtrace kurzorů ze seznamu křivek, neefektivní pohyb kurzorů, závislost na grafu. Proto musela být jejich implementace přepracována tak, aby nebyly závislé na grafu. Pro implementaci kurzorů byly použity samostatné objekty typu *Panel*, které jsou nezávislé na grafu. Byl vytvořen dialog s nastavením jednotlivých parametrů prohlížeče – grafu, os, tabulky signálů, kurzorů. Důležitým krokem bylo vytvoření algoritmu pro zoomování, protože zoomování obsažené v komponentě *ZedGraph* je pro křivky s velkým počtem vzorků nepoužitelné. Statistické a matematické funkce byly vyzkoušeny pouze na vytvořené testovací aplikaci s pevně danými křivkami. Testovací aplikace umožňovala zjistit minimum, maximum a střední hodnotu, dále byly implementovány jednoduché matematické funkce – sčítání, odečítání, násobení a dělení, a to jak mezi křivkami, tak mezi křivkou a zadaným číslem.

Posledním bodem bylo stanovení limit aplikace. Většina omezení byla vyřčena již během návrhu. Tato omezení se týkala hlavně počtu os X (maximálně 2), os Y (maximálně 8) a kurzorů (maximálně 2 vertikální a 2 horizontální). Dále bylo vyzkoušeno, že je aplikace schopna funkce i se signály s velkým počtem vzorků na křivku (až jeden milión vzorků).

7 Použitá literatura

[1] KAČMÁŘ, Dalibor. *Programujeme .NET aplikace ve Visual Studiu .NET*. Praha: Computer Press, 2001. 335 s. ISBN 80-7226-569-5.

[2] HERNANDEZ, Michael J.; VIECAS, John L. *Myslíme v jazyku SQL – Tvorba dotazu*. Praha: Grada Publishing, 2004. 380 s. ISBN 80-247-0899-X.

[3] VIRIUS, Miroslav. *C# pro zelenáče*. Praha: Neocortex, 2002. 256 s. ISBN 80-86330-11-7.

[4] SHARP, John. *Microsoft Visual C# 2008 Krok za krokem*. Praha: Computer Press, 2008. 592 s. ISBN 978-80-251-2027-9.

[5] NAGEL, Christian, et al. *C# 2008:Programujeme profesionálně*. Praha: Computer Press, 2009. 1904 s. ISBN 978-80-251-2401-7.

[6] YOUNG, Michael J. *XML krok za krokem*. Praha: Mobil Media, 2002. 471 s. ISBN 80-86593-28-2.

7.1 Internetové zdroje

[7] *.NET Framework Class Library* [online]. c2011, poslední změna v prosinci 2009 [cit. 2011-12-20]. Dostupné z: <http://msdn.microsoft.com/en-us/library/ms229335%28v=VS.90%29.aspx>.

[8] *ICS GmbH. TrendAnalyzer 3.0, Documentation and Mathematical Analysis of Measured Data Archives, Manual* [online]. [s.l.] : [s.n.], c2010 [cit. 2011-12-20]. Dostupné z: <http://www.icsgmbh.com/trviewer/tra30doceng.pdf>.

[9] *Future Design Controls* [online]. c2011 [cit. 2011-12-20]. Orion-M Data Viewer. Dostupné z: <http://www.futuredesigncontrols.com/Orion-M_Data_Viewer.HTM>.

[10] *iba AG. IbaAnalyzer – Analysis Program, Manual, Version 4.3* [online]. [s.l.] : [s.n.], c2009 [cit. 2011-12-20]. Dostupné z: <http://www.iba-ag.com/download/download.php?ID=1965&DownloadID=4-1118&type=SOFTWARE%20MANUAL&group=&product=&lang=>.

[11] *Procesní vizualizační systém SIMATIC WinCC* [online]. c2011 [cit. 2011-12-20]. Dostupné z: <http://www1.siemens.cz/ad/current/index.php?ctxnh=525faea5c2>.

[12] *SharpDevelop @ic#code* [online]. c2011 [cit. 2011-12-20]. Dostupné z: <http://www.icsharpcode.net/>.

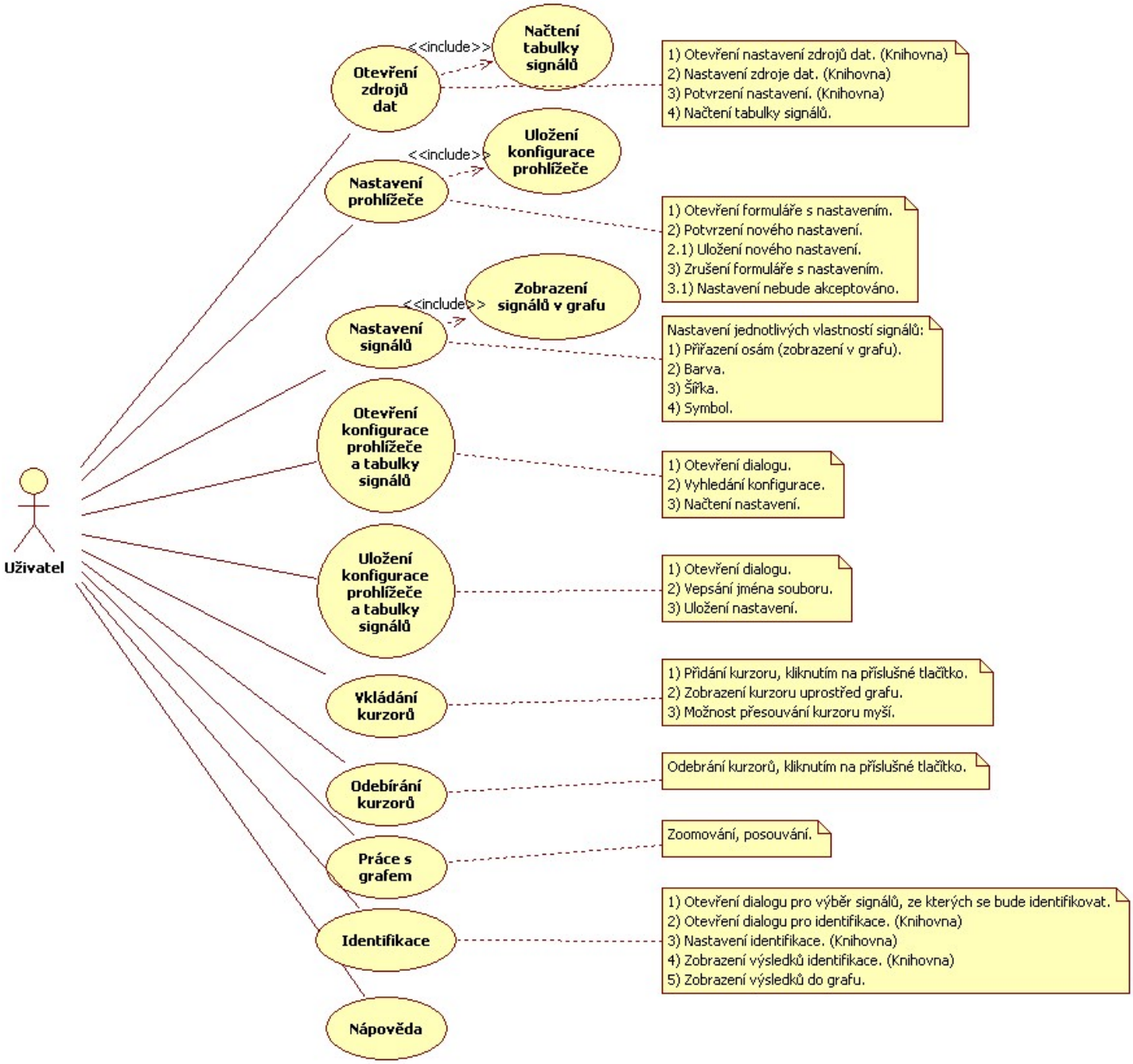
[13] *ZedGraph | Free Science & Engineering software downloads at SourceForge.net* [online]. c2011 [cit. 2011-12-20]. Dostupné z: <http://sourceforge.net/projects/zedgraph/>.

[14] *SourceGrid* [online]. c2011 [cit. 2011-12-20]. Dostupné z: <http://sourcegrid.codeplex.com/>.

[15] *DockPanel Suite | Free Science & Engineering software downloads at SourceForge.net* [online]. c2011 [cit. 2011-12-20]. Dostupné z: <http://sourceforge.net/projects/dockpanelsuite/>.

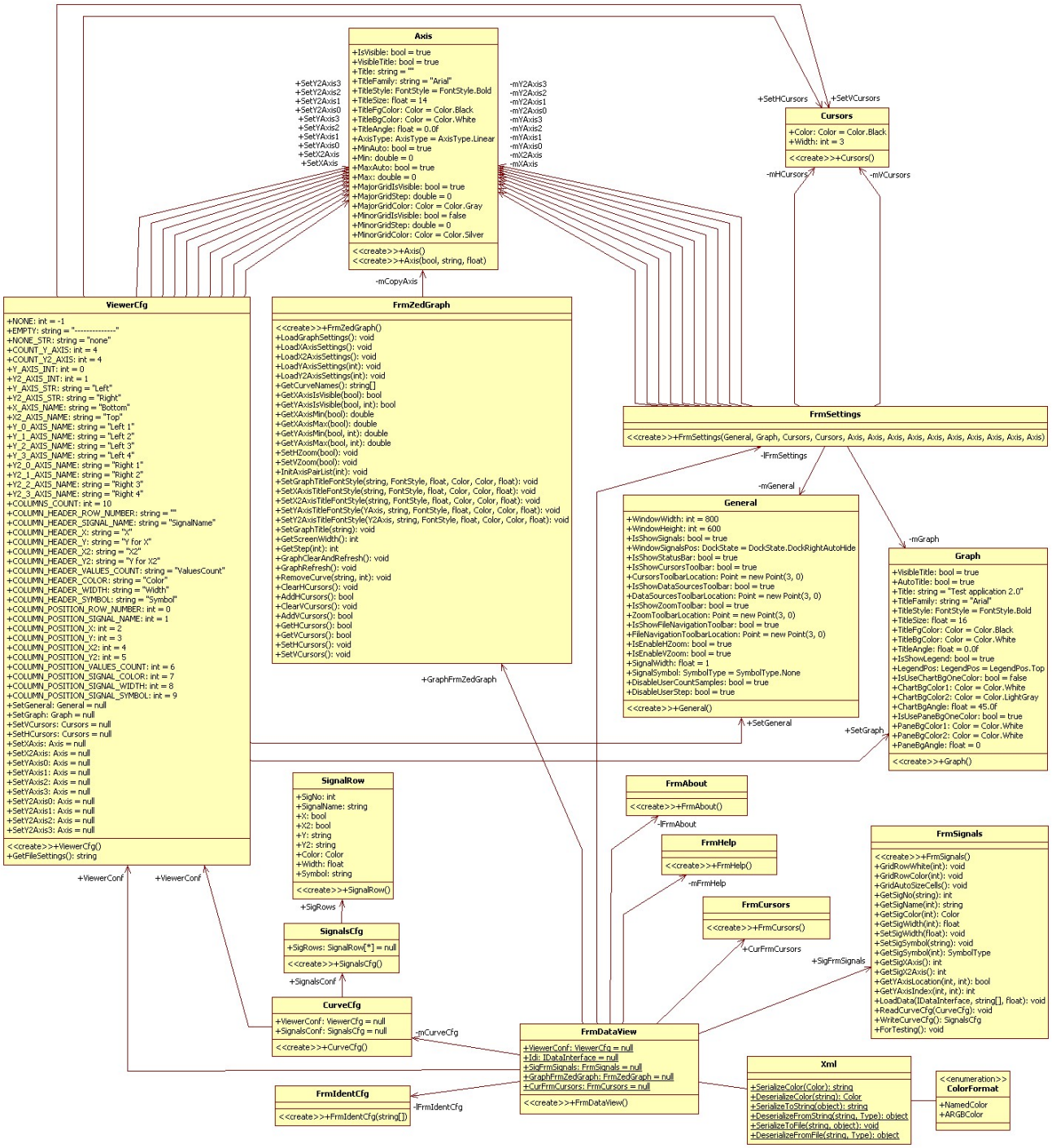
8 Seznam příloh

A Diagram případu užití



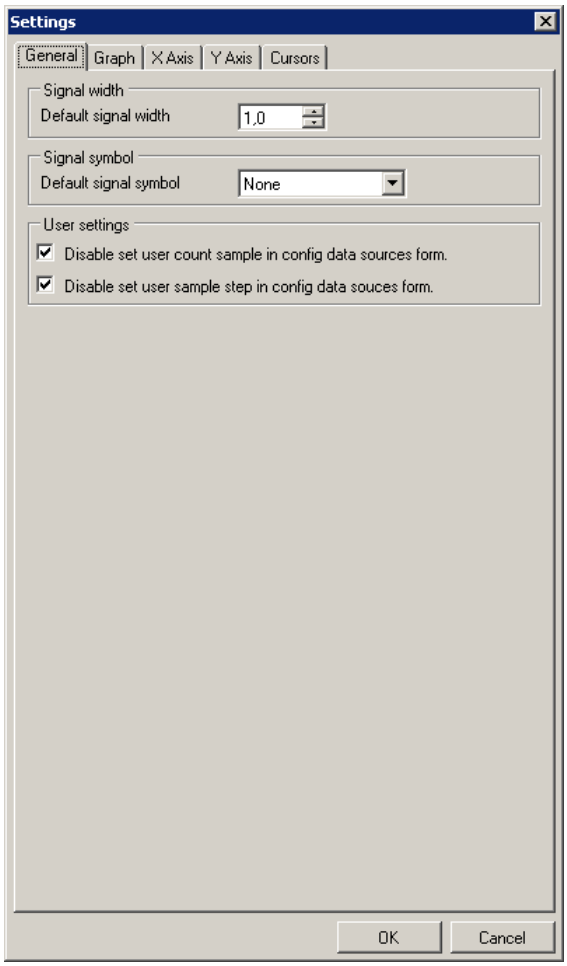
Obr. A.1: Diagram případu užití

B Diagram tříd

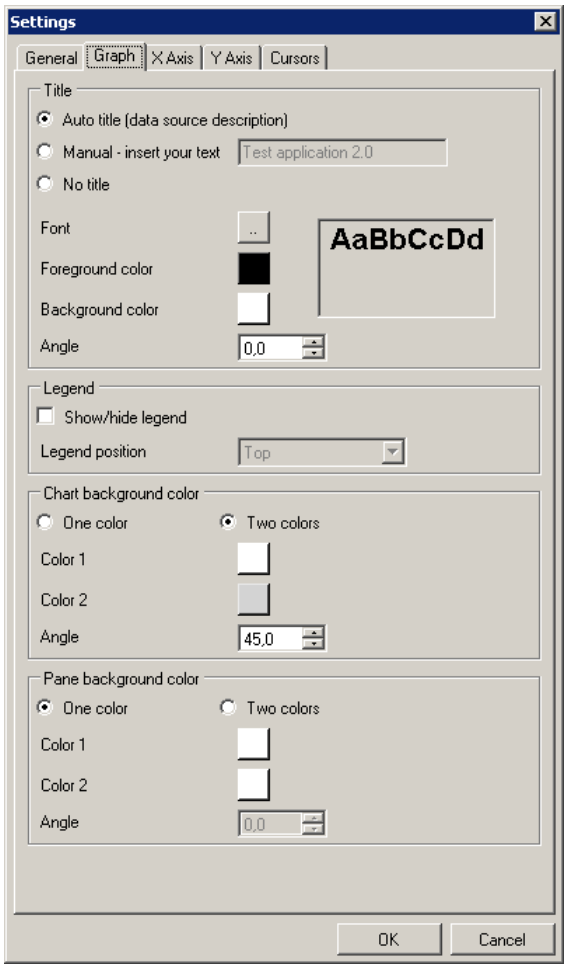


Obr. B.1: Diagram tříd vygenerovaný ze zdrojových kódu aplikace

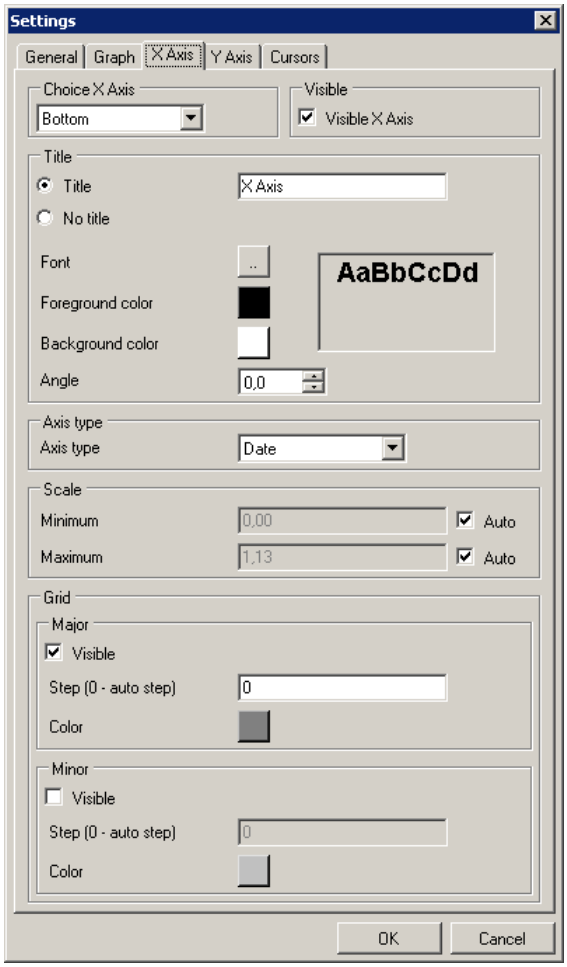
C Formuláře s nastavením prohlížeče



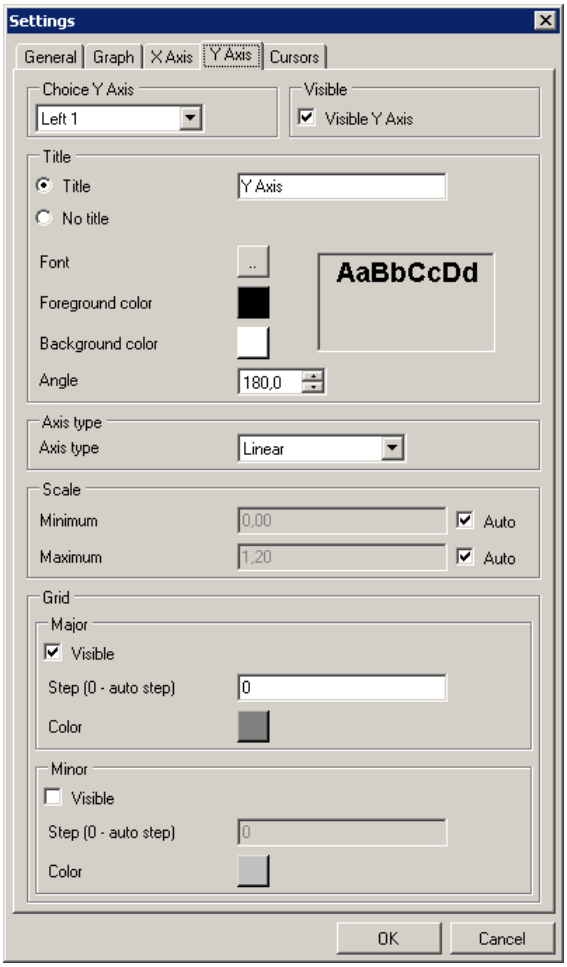
Obr. C.1: Hlavní nastavení



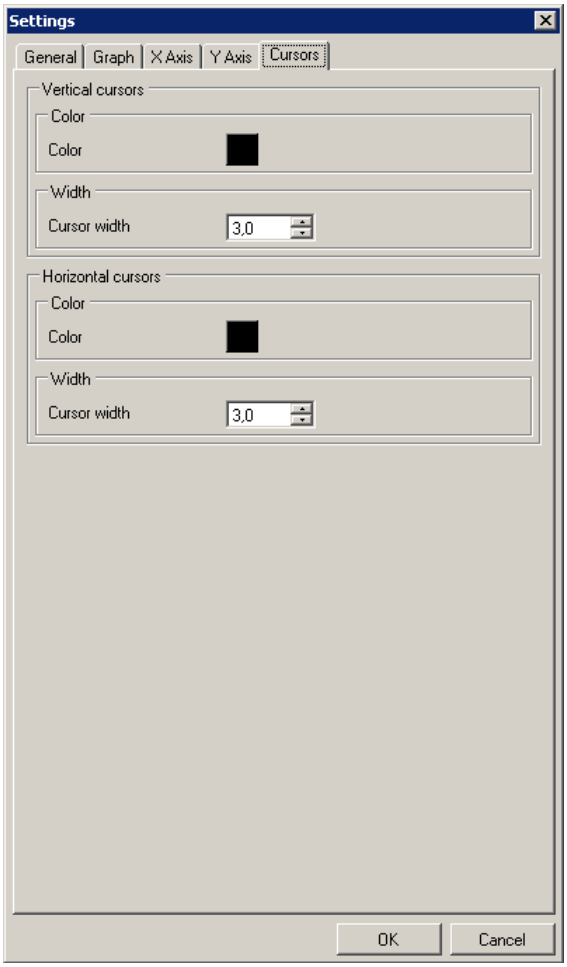
Obr. C.2: Nastavení grafu



Obr. C.3: Nastavení os X



Obr. C.4: Nastavení os Y



Obr. C.5: Nastavení kurzorů

D Obsah CD

Zde je popsán obsah přiloženého CD, obsah jednotlivých adresářů.

- ***CSharp_komponenty*** – obsahuje DLL knihovny použitých komponent:
  - *DockPanel Suite* verze 2.5.
  - *SourceGrid* verze 4.30.
  - *ZedGraph* verze 5.15.
- ***Diagramy_obrazky*** – obsahuje diagramy a obrázky použité v diplomové práci.
- ***Diplomova_prace*** – obsahuje PDF s obsahem této práce.
- ***Programy*** – obsahuje instalátory použitých programů:
  - *.NET Framework* verze 3.5.
  - *SharpDevelop* verze 3.2.
  - *StarUML* verze 5.0.
- ***Trendovy_prohlizec*** – vytvořená spustitelná aplikace.